

Classical Planning

willie

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

But first...

Konane

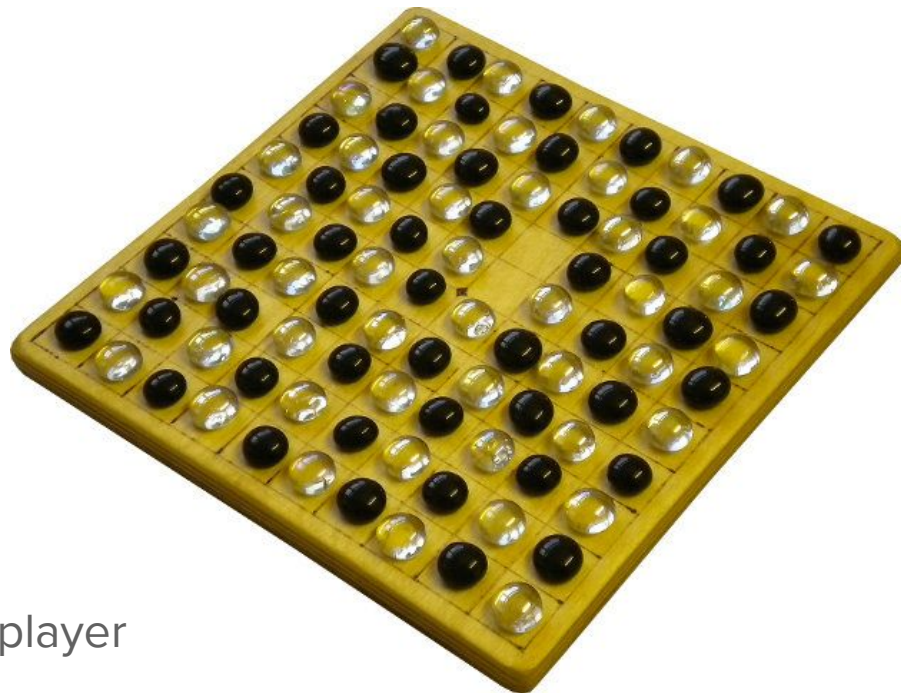
Hawaiian checkers

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

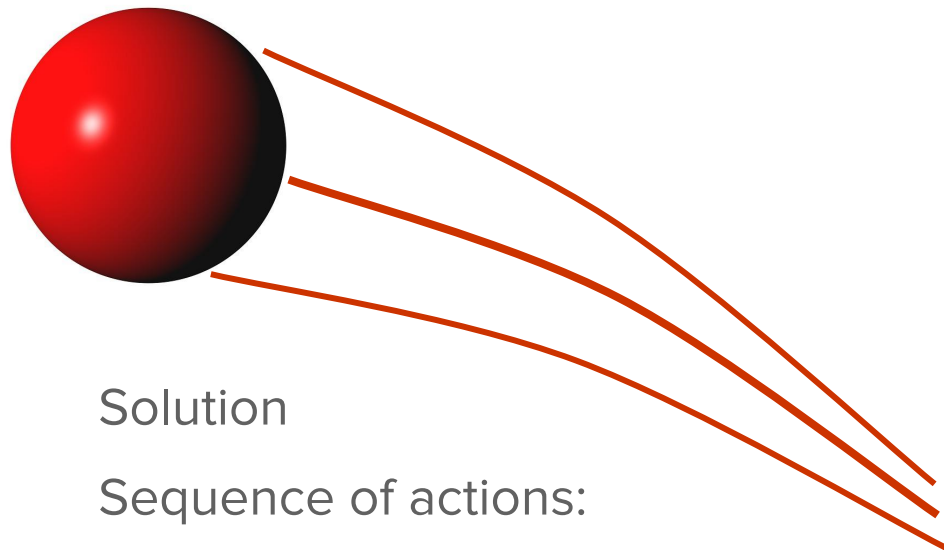
Get the ball to me (Willie)

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:

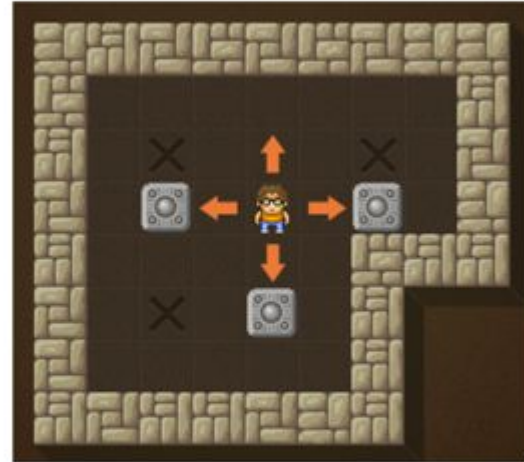
- Initial state
- Goal
- Available actions



Sokoban

Define the problem:

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



Sokoban

Define the problem:

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



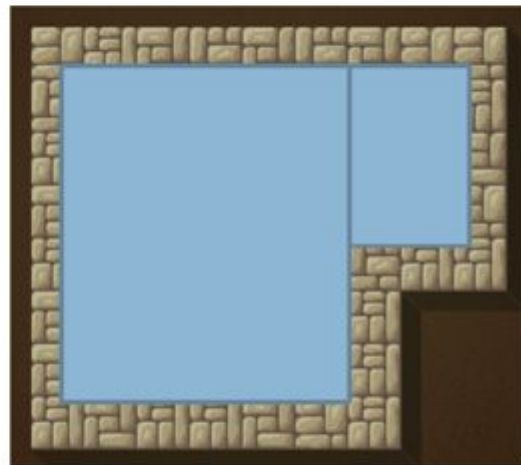
Find:

- 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)



Hubble Space Telescope

FGS

Hubble has three fine guidance sensors. Two are needed to point and lock the telescope on target, while the third can be used for astrometry, the precise measurement of stellar positions.

Primary mirror

Hubble's primary mirror is 7.8 feet (2.4 meters) in diameter. It is made of a special glass coated with aluminum and a compound that reflects ultraviolet light. It collects light from the telescope's targets and reflects it to the secondary mirror.

Secondary mirror

Like the primary mirror, Hubble's secondary mirror is made of special glass coated with aluminum and a compound to reflect ultraviolet light. It is 12 inches (30.5 centimeters) in diameter and reflects the light back through a hole in the primary mirror and into the instruments.

Aperture door

Hubble's aperture door can close, if necessary, to prevent light from the Sun from entering and potentially damaging the telescope or its instruments.

Communication antennas

Digital images and spectra stored in Hubble's solid-state recorders are converted to radio waves and then beamed through one of the spacecraft's high-gain antennas (HGAs) to a NASA communications satellite, which relays them to the ground. Because the HGAs would extend off the page above and below the spacecraft image, they are shown here pressed against the side of the telescope in their "bent positions." This is how they were configured at launch.

Solar panels

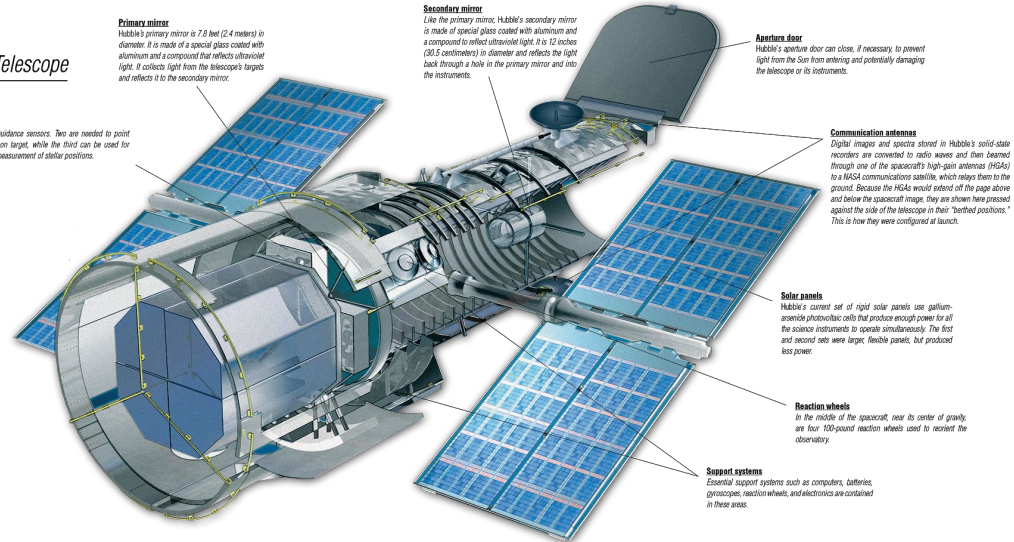
Hubble's current set of rigid solar panels use gallium-arsenide photovoltaic cells that produce enough power for all the science instruments to operate simultaneously. The first and second sets were larger, flexible panels, but produced less power.

Reaction wheels

In the middle of the spacecraft, near its center of gravity, are four 130-pound reaction wheels used to rotate the observatory.

Support systems

Essential support systems such as computers, batteries, gyroscopes, reaction wheels, and electronics are contained in these areas.

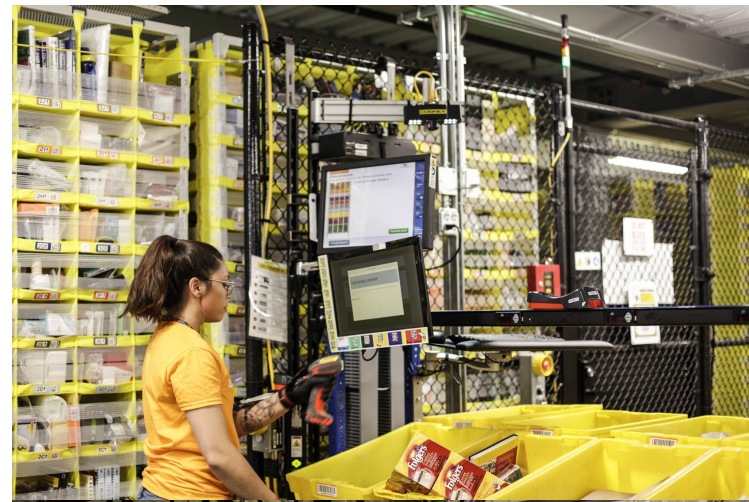


Amazon Robotics

Purchased Kiva in 2012

Warehouse robots

Planning to bring items to associates to pick customers orders

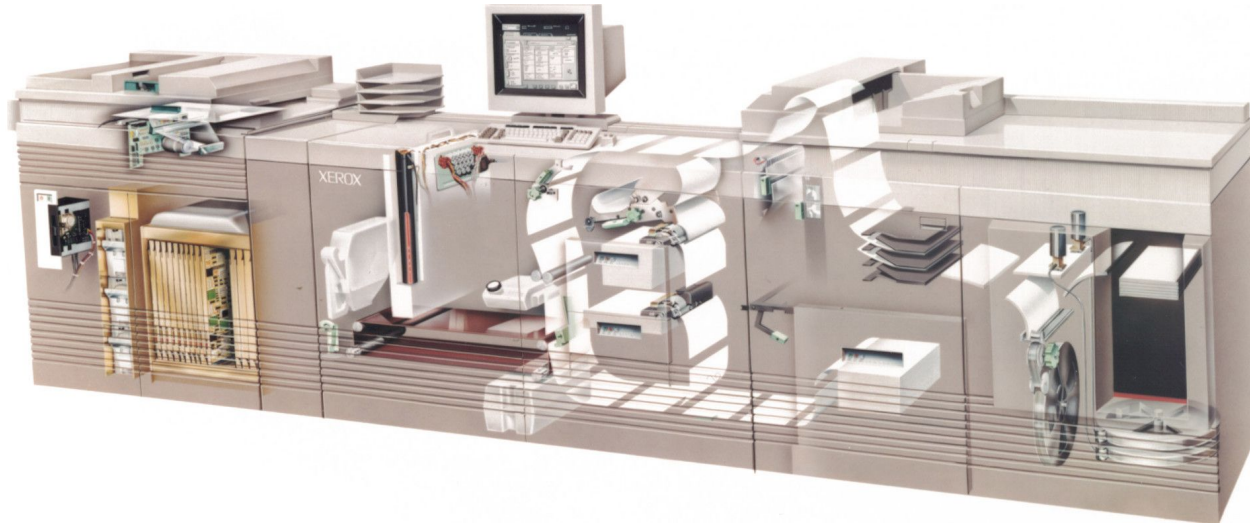


Xerox Printers

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 \leftrightarrow Planning

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

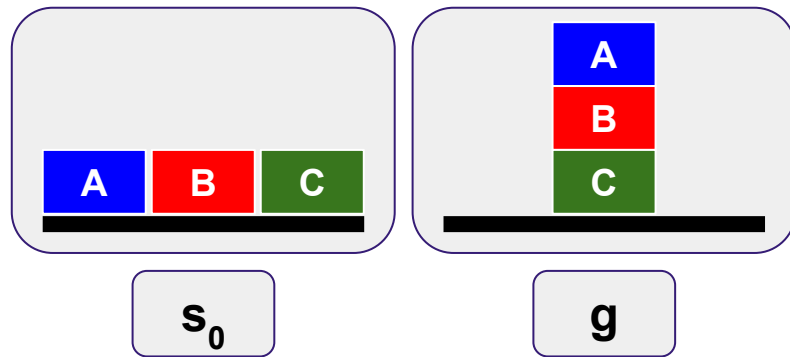
Situation Calculus

Logic for reasoning about changes in the state of the world

Prove: $\text{On}(A,B) \wedge \text{On}(B,C)$

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

$\text{On}(A, \text{Table}) \rightarrow \text{On}(A, \text{Table}, s_0)$

Atemporal functions and predicates:

True in any situation

$\text{red}(B)$

Situation Calculus

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

Possibility Axiom: preconditions \Rightarrow Poss(a,s)

$$\forall x \forall y \forall z \forall s \text{ On}(x, y, s) \wedge \text{Clear}(x, s) \wedge \text{Clear}(z, s) \Rightarrow \text{Poss}(\text{Move}(x, y, z), s)$$

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

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

Frame Problem

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

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

Frame Axioms: Inform the system about preserved relations

$$\forall s \ \forall x \ \forall y \ \forall z [\text{On}(x, y, s) \wedge (x \neq z)] \Rightarrow \text{On}(x, y, \text{Result}(\text{Move}(\text{Table}, z), s))$$

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

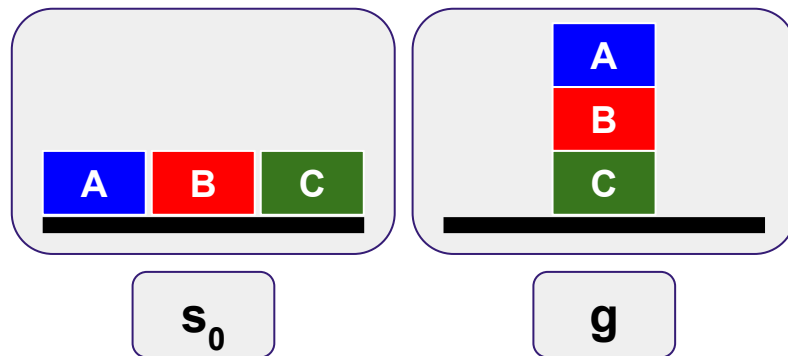
Blocks World

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



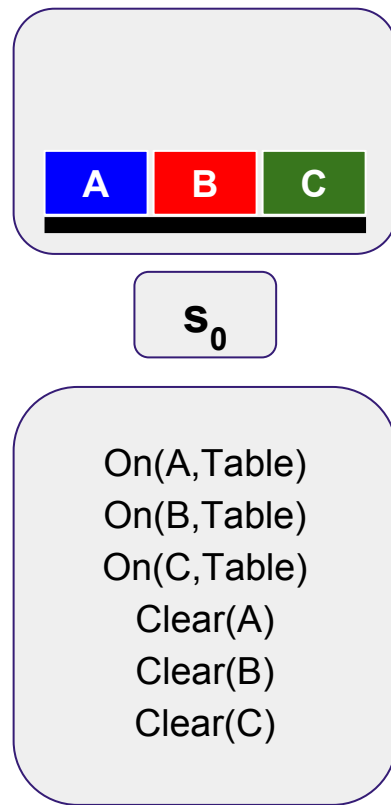
Blocks World

Initial state

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

Two relations:

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



Blocks World

Initial state

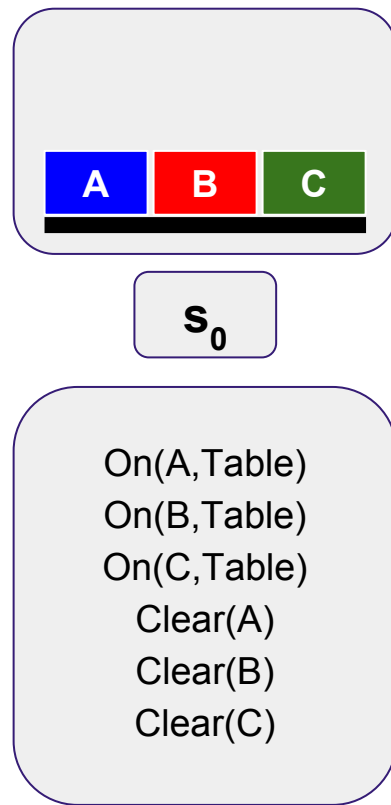
Representation of the properties of the domain using first-order logic literals

Ground and function-free

Completely specified

based on a **closed-world assumption**

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



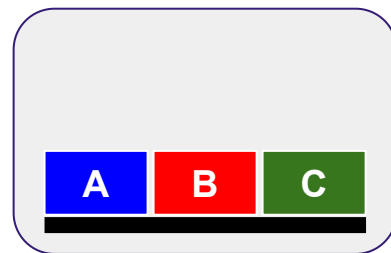
Blocks World

Initial state

Closed-world assumption

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

$\neg \text{On}(A,A)$
 $\neg \text{On}(A,B)$
 $\neg \text{On}(A,C)$
 $\neg \text{On}(B,A)$
 $\neg \text{On}(B,B)$
 $\neg \text{On}(B,C)$
 $\neg \text{On}(C,A)$
 $\neg \text{On}(C,B)$
 $\neg \text{On}(C,C)$
 $\neg \text{On}(A,A)$
 $\neg \text{On}(A,B)$
 $\neg \text{On}(A,C)$
 $\neg \text{On}(\text{Table},A)$
 $\neg \text{On}(\text{Table},B)$
 $\neg \text{On}(\text{Table},C)$
 $\neg \text{On}(\text{Table},\text{Table})$
 $)$
 $\neg \text{Clear}(\text{Table})$



s_0

$\text{On}(A,\text{Table})$
 $\text{On}(B,\text{Table})$
 $\text{On}(C,\text{Table})$
 $\text{Clear}(A)$
 $\text{Clear}(B)$
 $\text{Clear}(C)$

Blocks World

Goal

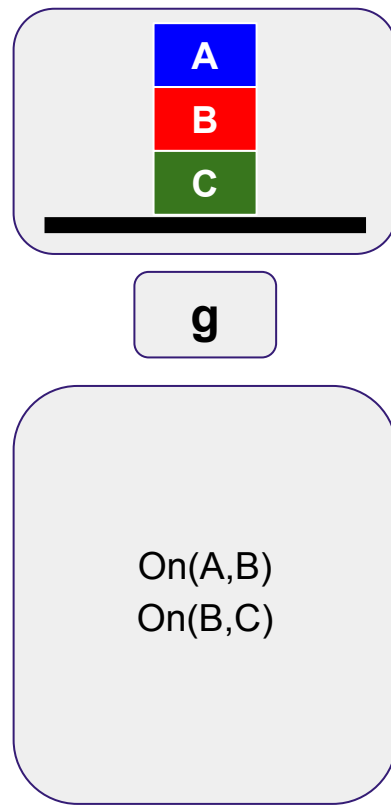
Representation of the properties of the domain using first-order logic literals

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**)

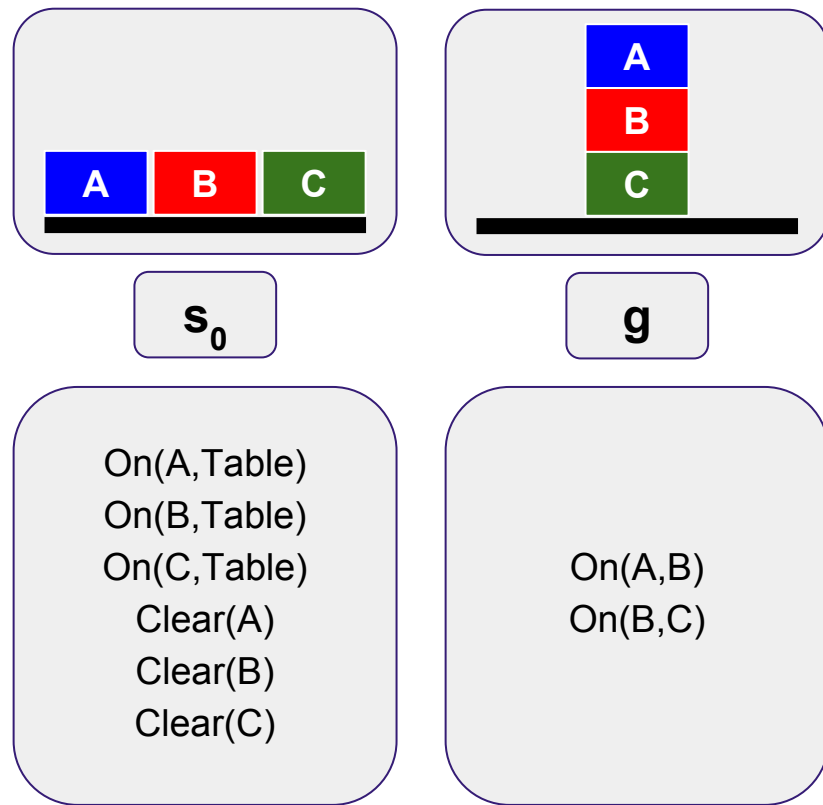


Blocks World

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

- Ground
- Function-free
- Positive

(list of literals forms a logical conjunction)

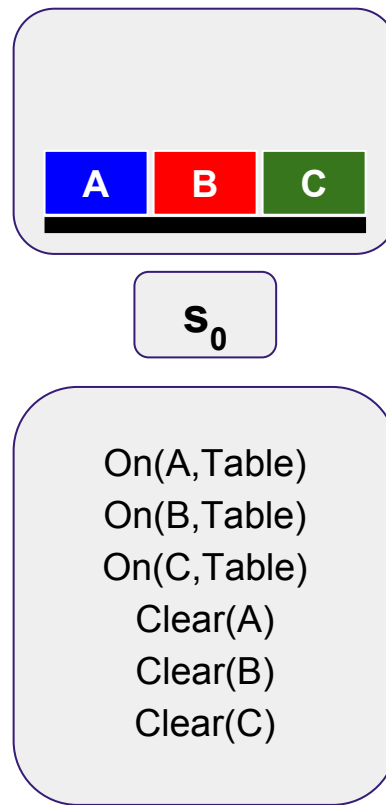


Blocks World

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

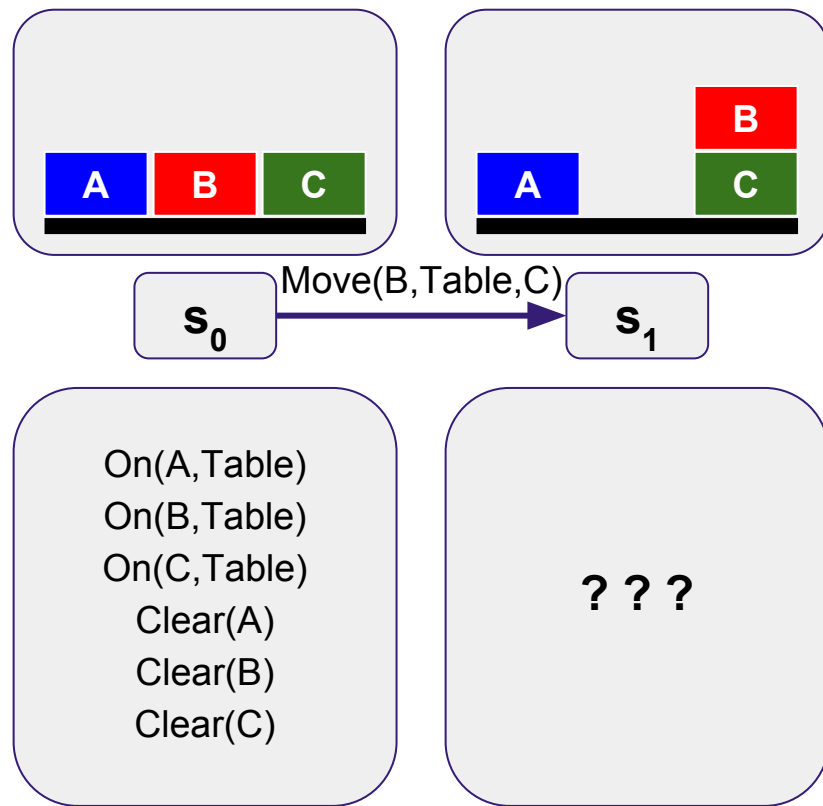


Blocks World

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

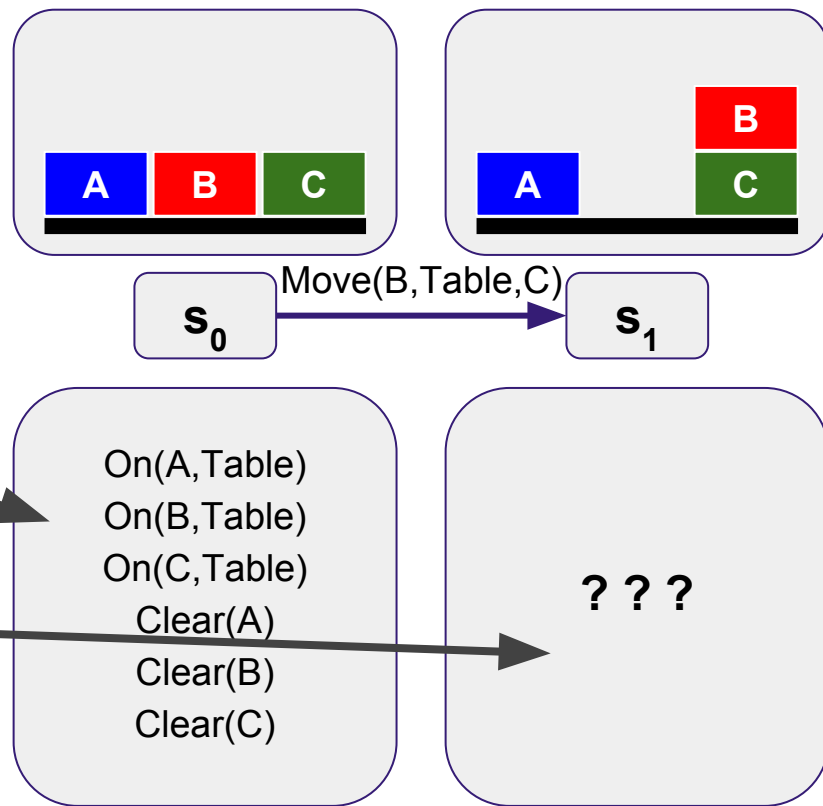
$\text{Move}(b, x, y)$

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

$\text{Move}(b, x, y)$

Preconditions

$\text{On}(b, x)$

$\text{Clear}(b)$

$\text{Clear}(y)$

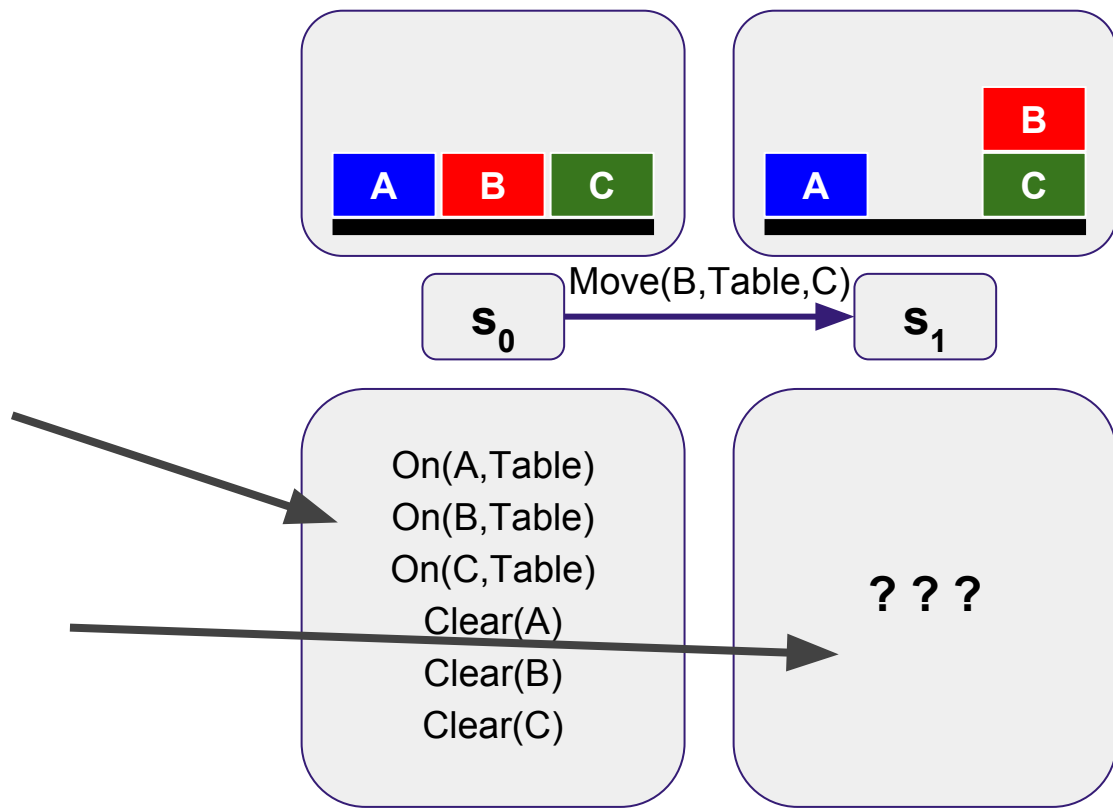
Effects

$\text{On}(b, y)$

$\text{Clear}(x)$

$\neg \text{On}(b, x)$

$\neg \text{Clear}(y)$



Blocks World

Available actions

$\text{Move}(B, \text{Table}, C)$

Preconditions

$\text{On}(B, \text{Table})$ ✓

$\text{Clear}(B)$ ✓

$\text{Clear}(C)$ ✓

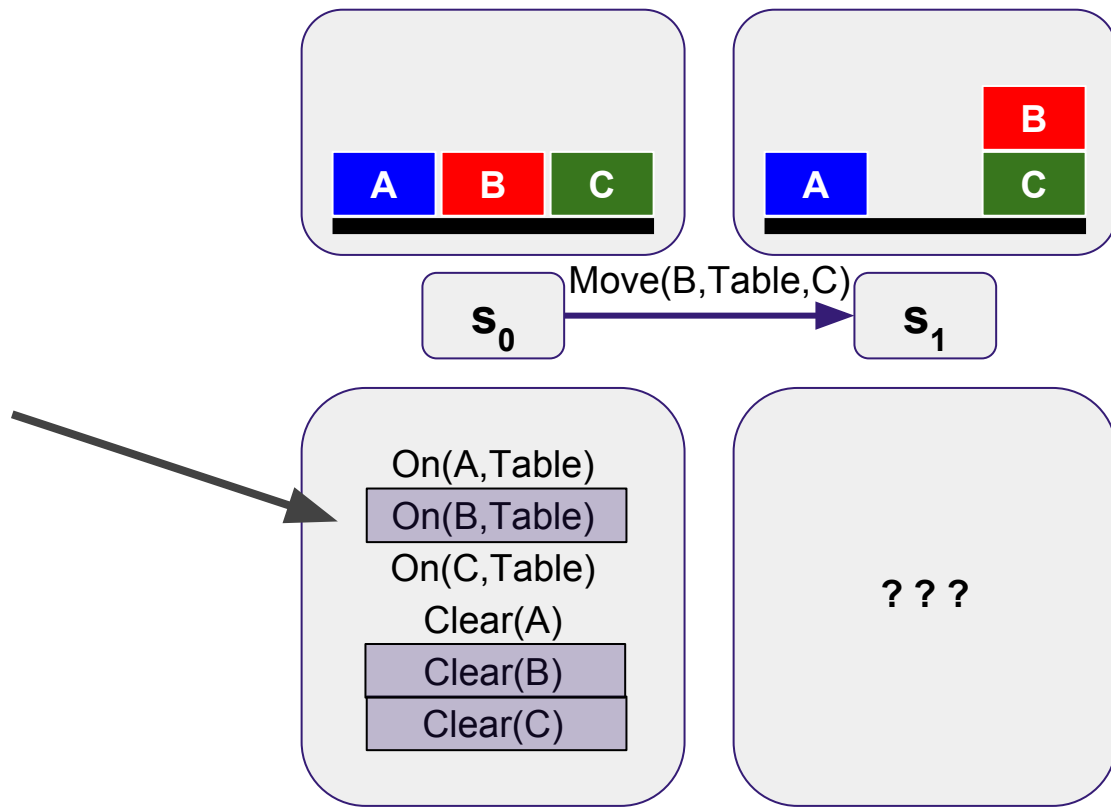
Effects

$\text{On}(b, y)$

$\text{Clear}(x)$

$\neg \text{On}(b, x)$

$\neg \text{Clear}(y)$



Blocks World

Available actions

$\text{Move}(B, \text{Table}, C)$

Preconditions

$\text{On}(B, \text{Table})$

$\text{Clear}(B)$

$\text{Clear}(C)$

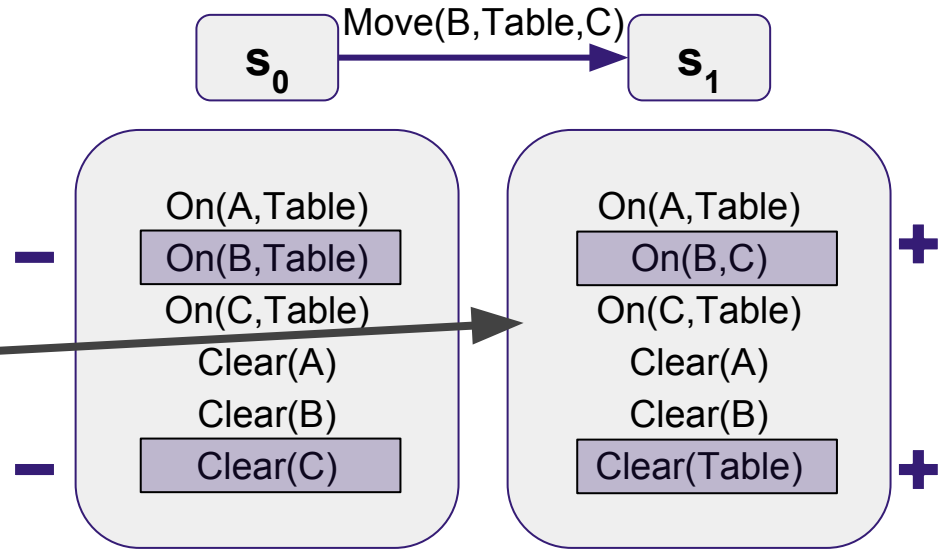
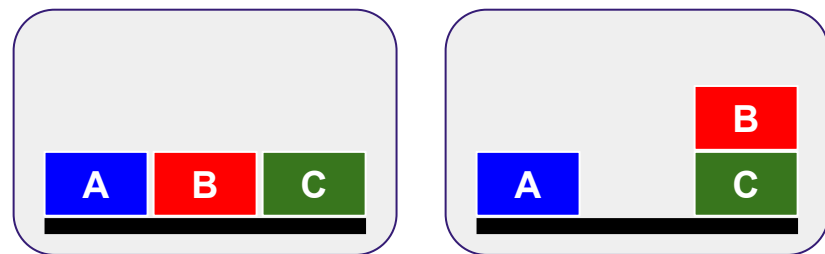
Effects

$+$ $\text{On}(b, y)$

$+$ $\text{Clear}(x)$

$-$ $\neg \text{On}(b, x)$

$-$ $\neg \text{Clear}(y)$



Blocks World

Available actions

$\text{Move}(b, x, y)$

Preconditions

$\text{On}(b, x)$

$\text{Clear}(b)$

$\text{Clear}(y)$

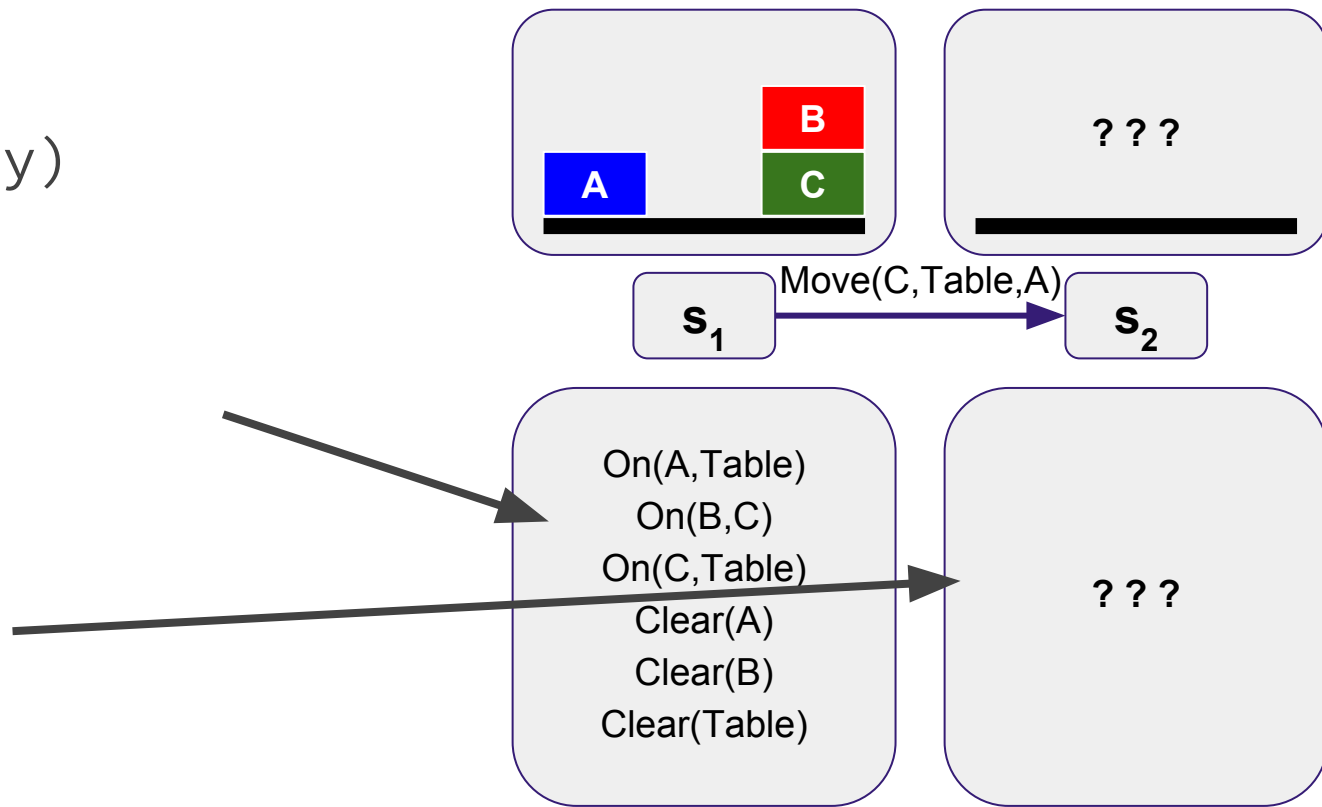
Effects

$\text{On}(b, y)$

$\text{Clear}(x)$

$\neg \text{On}(b, x)$

$\neg \text{Clear}(y)$



Blocks World

Available actions

$\text{Move}(C, \text{Table}, A)$

Preconditions

$\text{On}(C, \text{Table})$ ✓

$\text{Clear}(C)$ ✗

$\text{Clear}(A)$ ✓

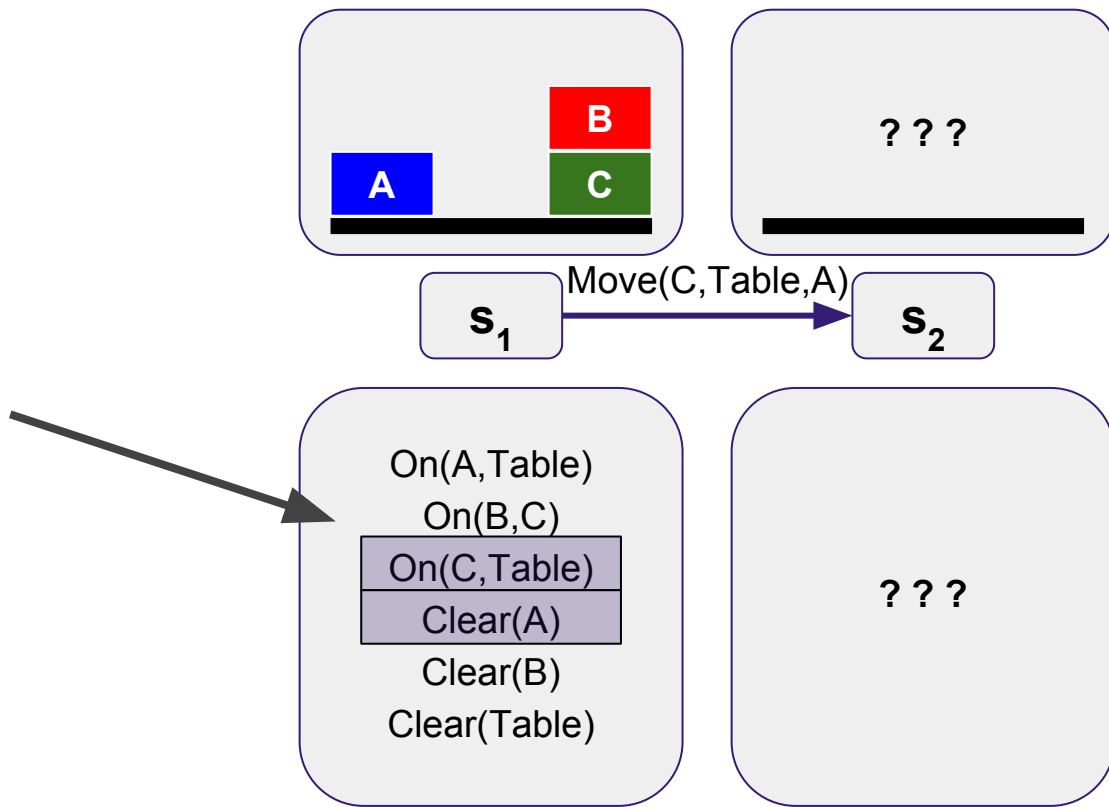
Effects

$\text{On}(C, A)$

$\text{Clear}(\text{Table})$

$\neg \text{On}(C, \text{Table})$

$\neg \text{Clear}(A)$



Blocks World

Available actions

$\text{Move}(A, \text{Table}, B)$

Preconditions

$\text{On}(A, \text{Table})$ ✓

$\text{Clear}(A)$ ✓

$\text{Clear}(B)$ ✓

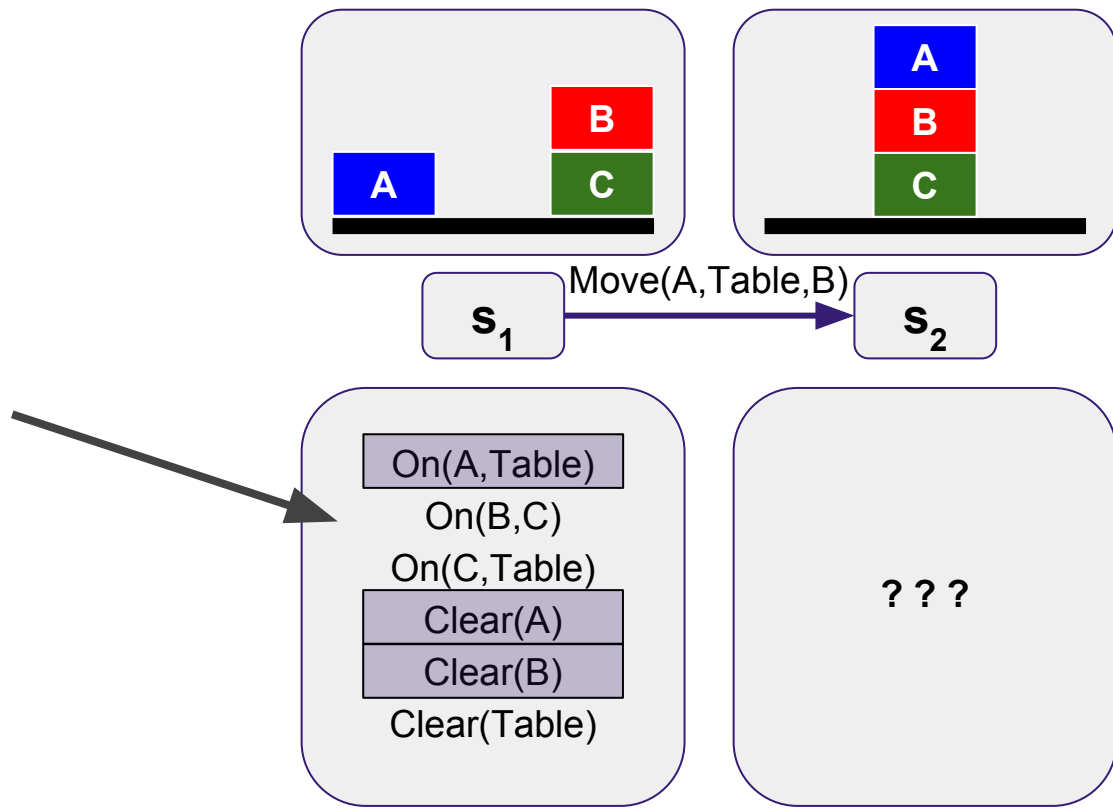
Effects

$\text{On}(A, B)$

$\text{Clear}(\text{Table})$

$\neg \text{On}(A, \text{Table})$

$\neg \text{Clear}(B)$



Blocks World

Available actions

$\text{Move}(A, \text{Table}, B)$

Preconditions

$\text{On}(A, \text{Table})$

$\text{Clear}(A)$

$\text{Clear}(B)$

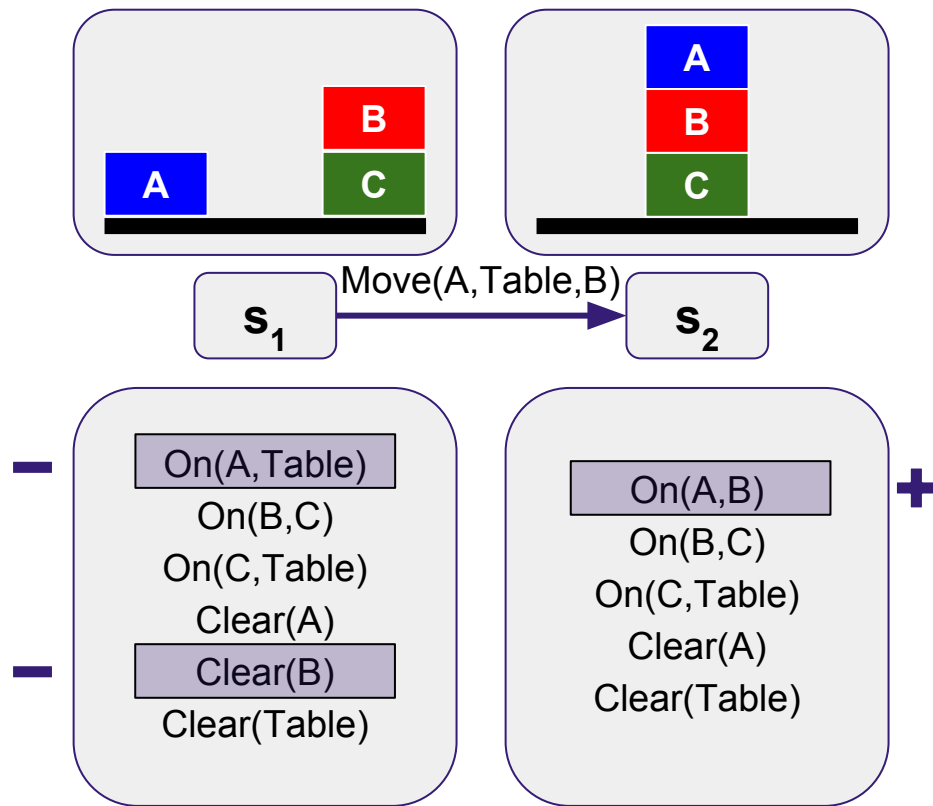
Effects

$+$ $\text{On}(A, B)$

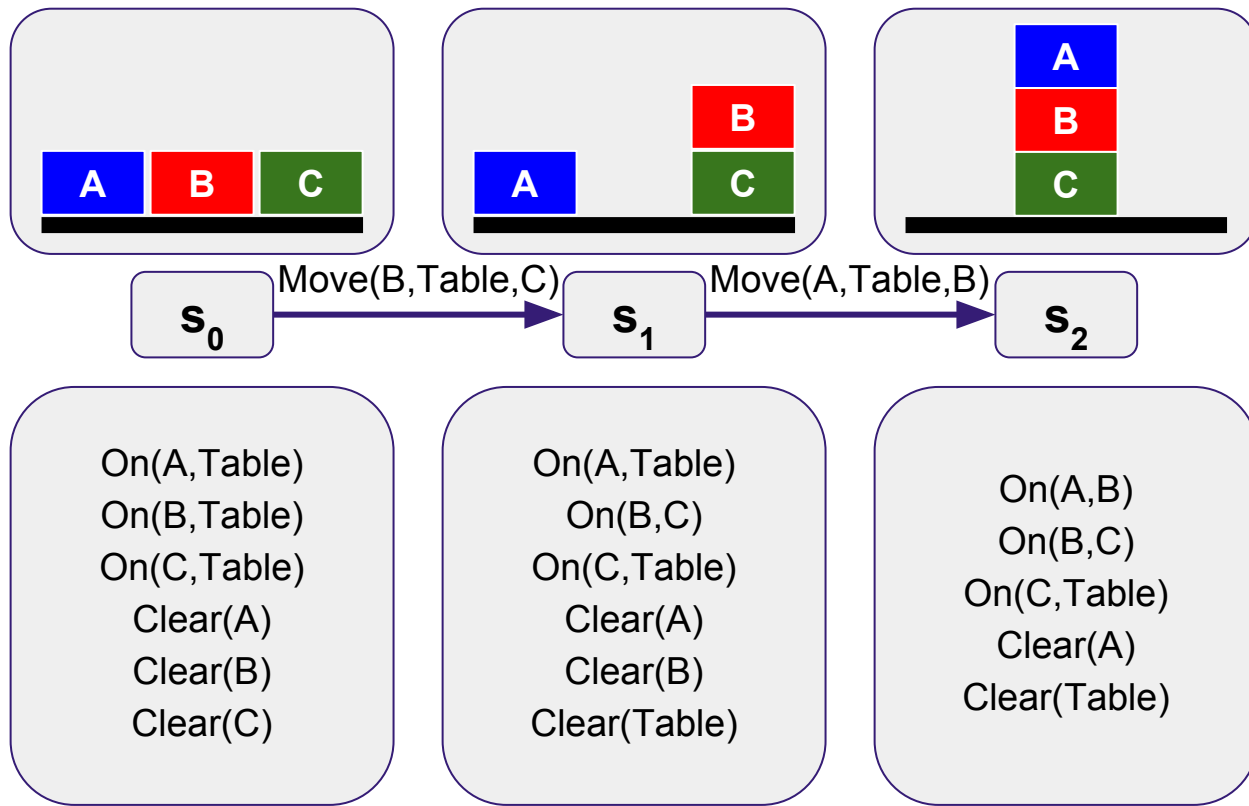
$+$ $\text{Clear}(\text{Table})$

$-$ $\neg \text{On}(A, \text{Table})$

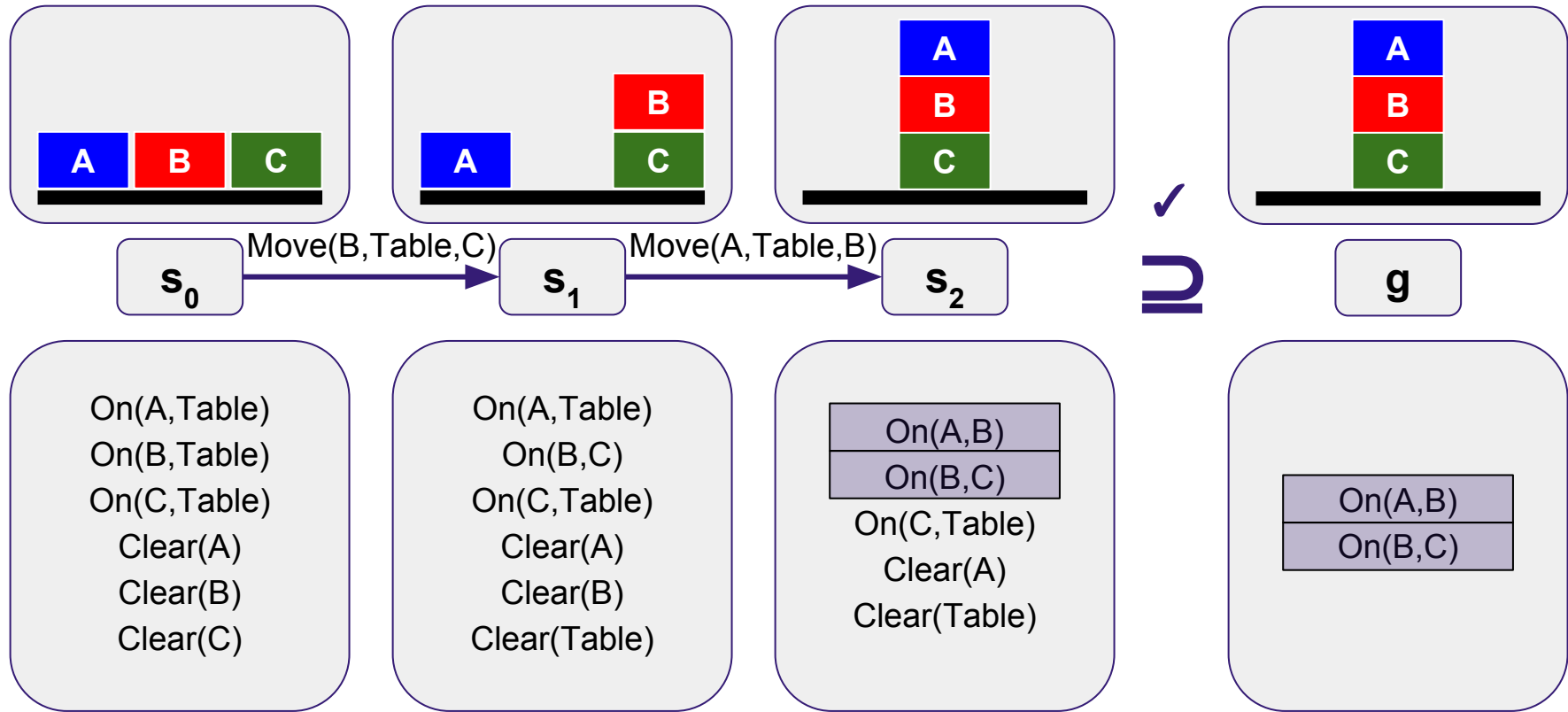
$-$ $\neg \text{Clear}(B)$



Blocks World



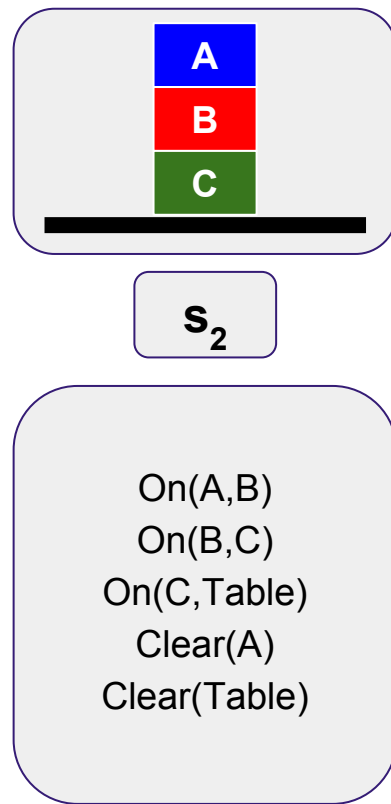
Blocks World



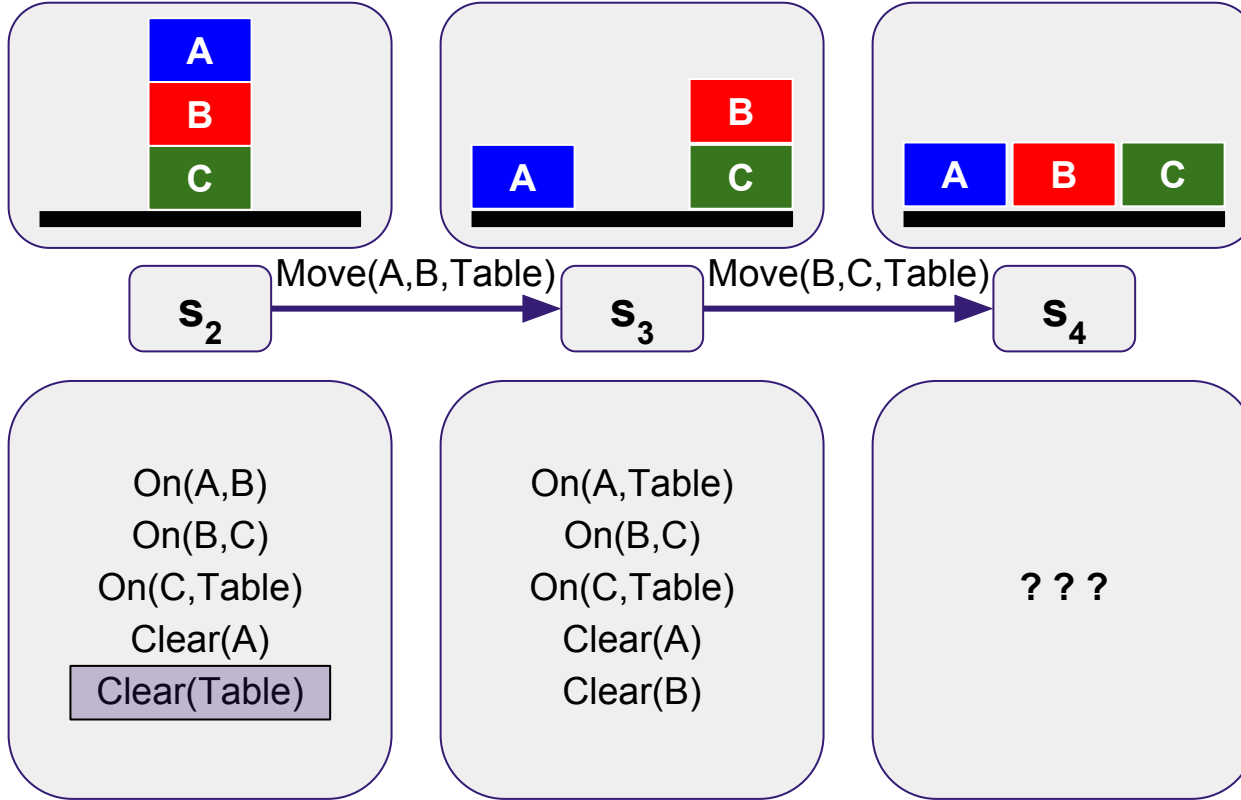
Blocks World

Is our representation of the blocks world correct?

Consider moving back blocks A and B back to the table



Blocks World



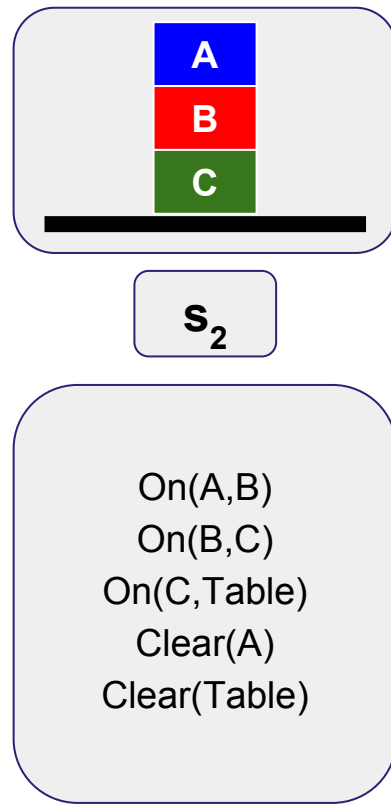
Blocks World

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)



STRIPS planning

Init($\text{On}(\text{A}, \text{Table}) \wedge \text{On}(\text{B}, \text{Table}) \wedge \text{On}(\text{C}, \text{Table}) \wedge \text{Clear}(\text{A}) \wedge \text{Clear}(\text{B}) \wedge \text{Clear}(\text{C})$)

Goal($\text{On}(\text{A}, \text{B}) \wedge \text{On}(\text{B}, \text{C})$)

Action(Move(b,x,y),

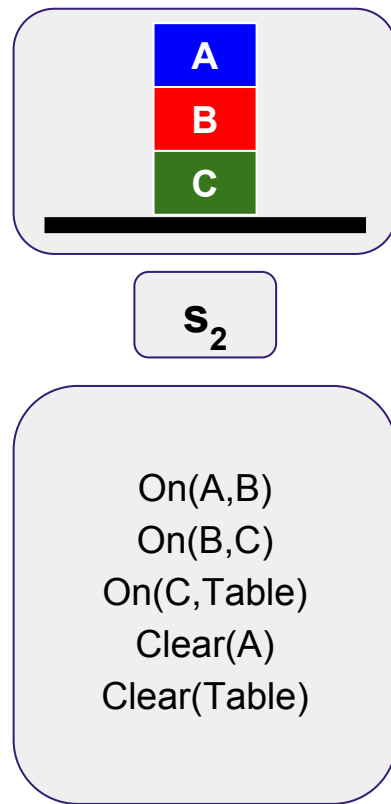
PRECONDITIONS: $\text{On}(\text{b}, \text{x}) \wedge \text{Clear}(\text{b}) \wedge \text{Clear}(\text{y})$

EFFECTS: $\text{On}(\text{b}, \text{y}) \wedge \text{Clear}(\text{x}) \wedge \neg \text{On}(\text{b}, \text{x}) \wedge \neg \text{Clear}(\text{y})$)

Action(MoveToTable(b,x),

PRECONDITIONS: $\text{On}(\text{b}, \text{x}) \wedge \text{Clear}(\text{b})$

EFFECTS: $\text{On}(\text{b}, \text{Table}) \wedge \text{Clear}(\text{x}) \wedge \neg \text{On}(\text{b}, \text{x})$)



STRIPS planning

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

Action(Move(b,x,y),

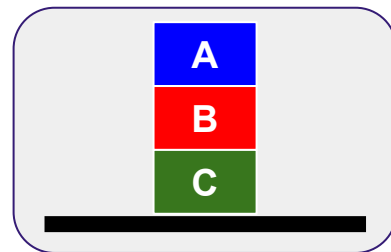
PRECONDITIONS: $\text{On}(b,x) \wedge \text{Clear}(b) \wedge \text{Clear}(y)$

EFFECTS: $\text{On}(b,y) \wedge \text{Clear}(x) \wedge \neg \text{On}(b,x) \wedge \neg \text{Clear}(y)$

Action(MoveToTable(b,x),

PRECONDITIONS: $\text{On}(b,x) \wedge \text{Clear}(b)$

EFFECTS: $\text{On}(b,\text{Table}) \wedge \text{Clear}(x) \wedge \neg \text{On}(b,x)$



s_2

$\text{On}(A,B)$
 $\text{On}(B,C)$
 $\text{On}(C,\text{Table})$
 $\text{Clear}(A)$
 $\text{Clear}(\text{Table})$

STRIPS planning

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

Action(Move(b,x,y),

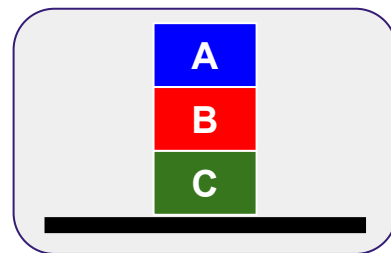
PRECONDITIONS: $\text{On}(b,x) \wedge \text{Clear}(b) \wedge \text{Clear}(y)$

EFFECTS: $\text{On}(b,y) \wedge \text{Clear}(x) \wedge \neg \text{On}(b,x) \wedge \neg \text{Clear}(y)$

Action(MoveToTable(b,x),

PRECONDITIONS: $\text{On}(b,x) \wedge \text{Clear}(b)$

EFFECTS: $\text{On}(b,\text{Table}) \wedge \text{Clear}(x) \wedge \neg \text{On}(b,x)$



s_2

$\text{On}(A,B)$
 $\text{On}(B,C)$
 $\text{On}(C,\text{Table})$
 $\text{Clear}(A)$
 $\text{Clear}(\text{Table})$

STRIPS planning

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

Action(Move(b,x,y),

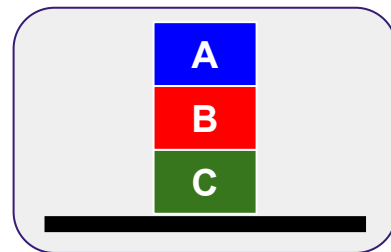
PRECONDITIONS: $\text{On}(b,x) \wedge \text{Clear}(b) \wedge \text{Clear}(y)$

EFFECTS: $\text{On}(b,y) \wedge \text{Clear}(x) \wedge \neg \text{On}(b,x) \wedge \neg \text{Clear}(y)$

Action(MoveToTable(b,x),

PRECONDITIONS: $\text{On}(b,x) \wedge \text{Clear}(b)$

EFFECTS: $\text{On}(b,\text{Table}) \wedge \text{Clear}(x) \wedge \neg \text{On}(b,x)$



s_2

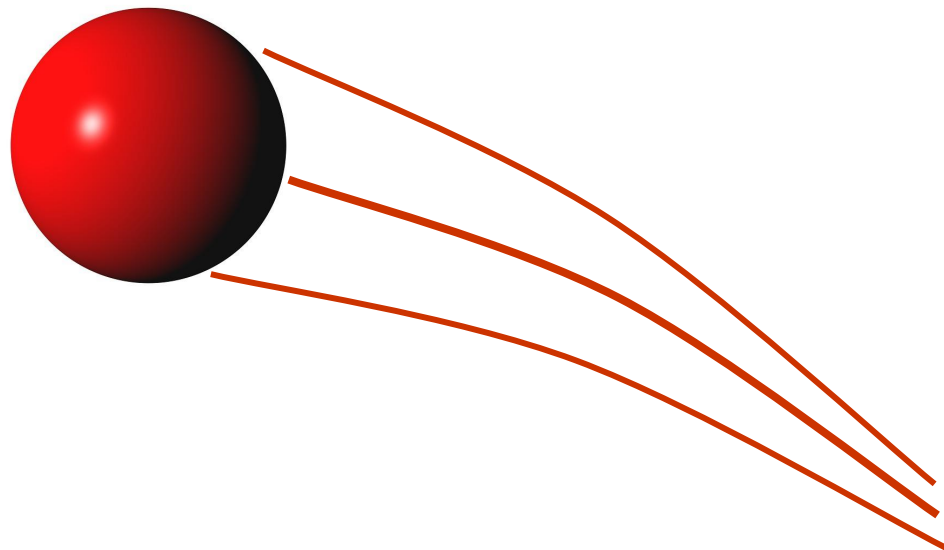
$\text{On}(A,B)$
 $\text{On}(B,C)$
 $\text{On}(C,\text{Table})$
 $\text{Clear}(A)$
 $\text{Clear}(\text{Table})$

Pass the ball - STRIPS

Init(HasBall(Ethan) \wedge
LastLetter(Ethan, n))

Goal(HasBall(Willie) \wedge
FirstLetter(Willie, w))

Action(PassBall(x,l,y,f)
PRECONDITIONS: LastLetter(x,l) \wedge FirstLetter(y,f) \wedge
Same(l,f) \wedge HasBall(x)
EFFECTS: HasBall(y) \wedge \neg HasBall(x)



Next time

Problem

Domain

Description

Language
