Stavros Vassos, University of Athens, Greece stavrosv@di.uoa.gr May 2012

INTRODUCTION TO AI STRIPS PLANNING

.. and Applications to Video-games!

Course overview

- Lecture 1: Game-inspired competitions for AI research, AI decision making for non-player characters in games
- Lecture 2: STRIPS planning, state-space search
- Lecture 3: Planning Domain Definition Language (PDDL), using an award winning planner to solve Sokoban
- Lecture 4: Planning graphs, domain independent heuristics for STRIPS planning
- Lecture 5: Employing STRIPS planning in games: SimpleFPS, iThinkUnity3D, SmartWorkersRTS
- Lecture 6: Planning beyond STRIPS

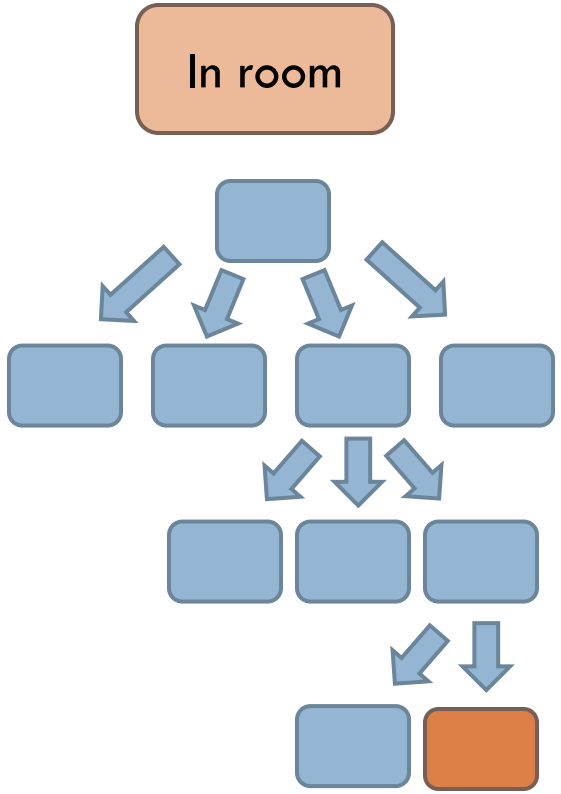
Goal Oriented Action Planning (GOAP)

- Video Games:
  - Finite State Machines
  - Decision Diagrams
  - Behavior Trees
  - Goal Oriented Action Planning
- Academic AI on agents:
  - Knowledge representation, First-order logic, Classical planning, Planning with preferences, …
  - Belief-Desire-Intention architecture, Agent-based programming, …
  - Probabilistic reasoning, Bayesian networks, Utility theory, Markov Decision Processes, …

Goal Oriented Action Planning (GOAP)

- **Search** for a **plan** that achieves the **goal** in the current state

Move into room	Preconditions: Door open Effects: In room
Move to door	Preconditions: - Effects: At door
Unlock door	Preconditions: Hold key Effects: Door open
Kick door	Preconditions: - Effects: Door open



Goal Oriented Action Planning (GOAP)

- **Search** for a **plan** that achieves the **goal** in the current state
- There is a **standard formalism** for representing such problems in **AI literature** and a **large body of work**!
- Video-games: The best strategy is to use the existing formalism and take advantage of the work in the literature

Course overview

- Lecture 1: Game-inspired competitions for AI research, AI decision making for non-player characters in games
- Lecture 2: **STRIPS planning**, state-space search
- Lecture 3: Planning Domain Definition Language (PDDL), using an award winning planner to solve Sokoban
- Lecture 4: Planning graphs, domain independent heuristics for STRIPS planning
- Lecture 5: Employing STRIPS planning in games: SimpleFPS, iThinkUnity3D, SmartWorkersRTS
- Lecture 6: Planning beyond STRIPS

STRIPS planning

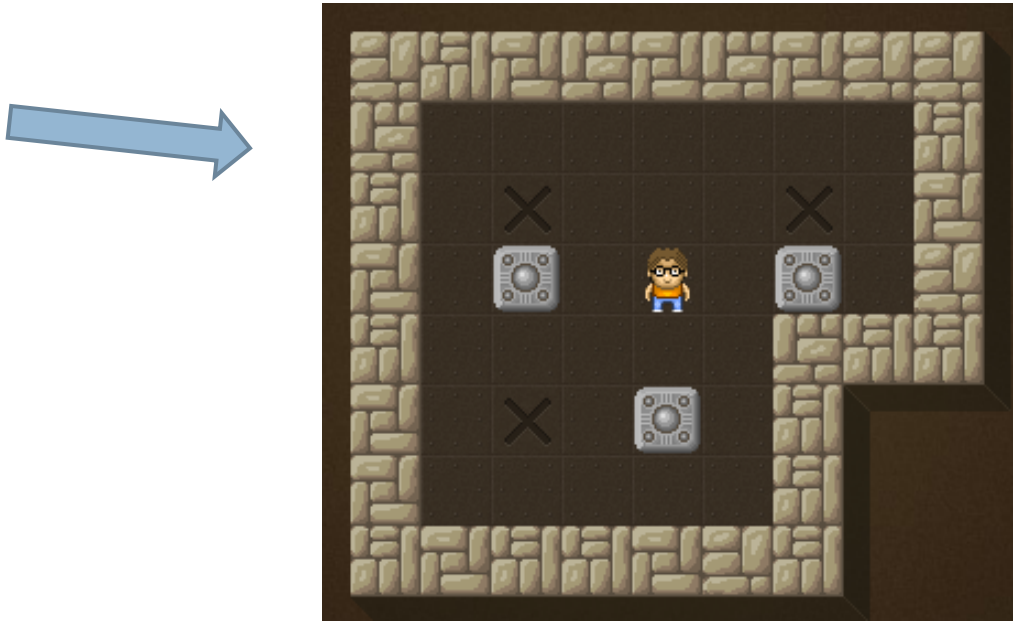
- Typical description of a planning problem:
  - Initial state
  - Goal
  - Available actions
- Typical solution of a planning problem:
  - A sequence of actions that when executed in the initial state results in a state that satisfies the goal condition.
- Planning: The automatic discovery of such solutions

What is a planning problem?

- Let's start with a simple example..

What is a planning problem?

- Given:
  - Initial state

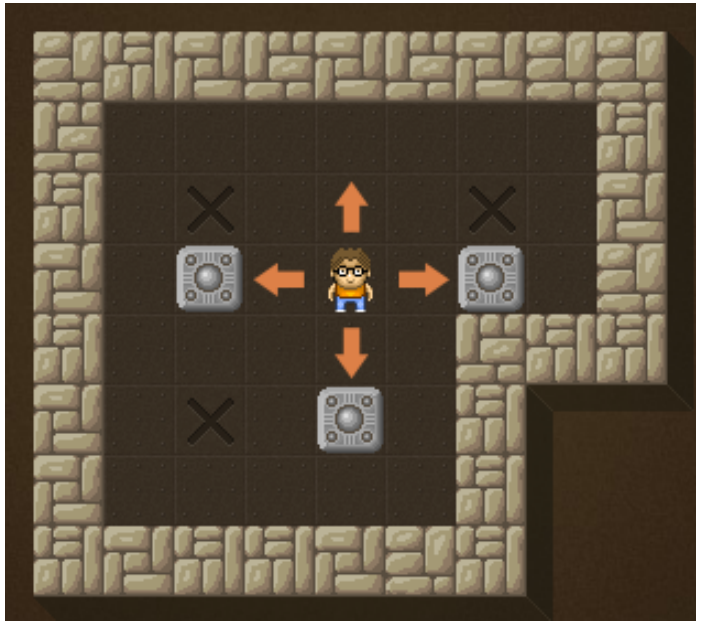


What is a planning problem?

- Given:
  - Initial state
  - Goal

What is a planning problem?

- Given:
  - Initial state
  - Goal
  - Available actions



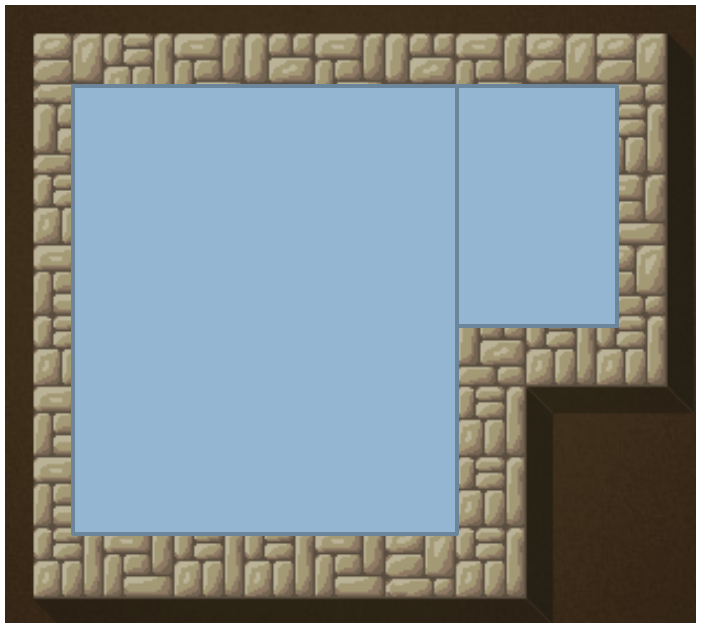
What is a planning problem?

- Given:
  - Initial state
  - Goal
  - Available actions

- Find:
  - A **sequence of actions** that achieves the goal
  - E.g.: **[Left, Down, Left, Up, …]**

What is a planning problem?

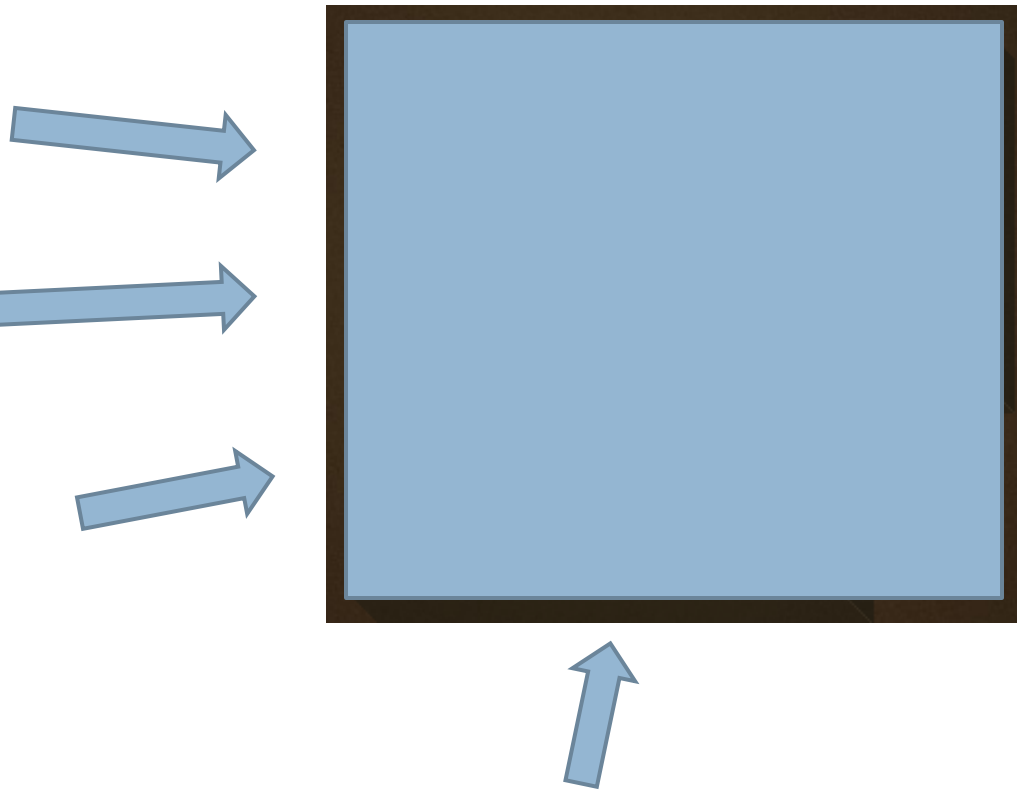
- Given:
  - Initial state
  - Goal
  - Available actions



- Find:
  - A **sequence of actions** that achieves the goal
  - A **method** that is capable of finding a solution **for every initial state and goal**

What is a planning problem?

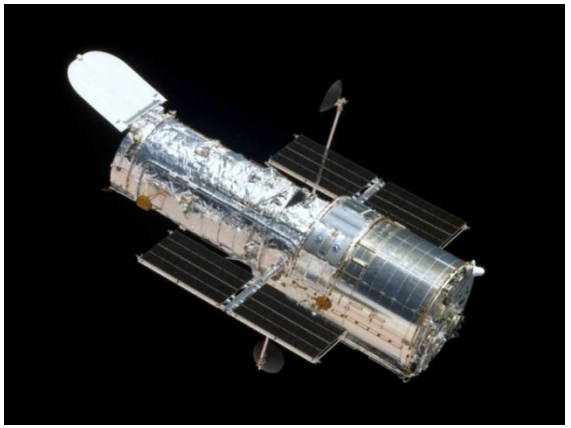
- Given:
  - Initial state
  - Goal
  - Available actions

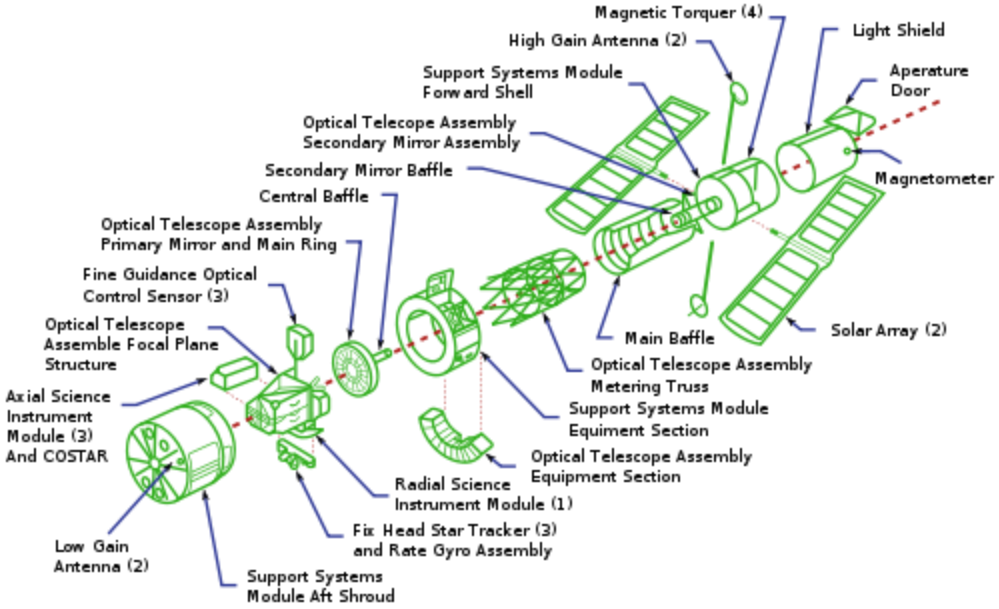


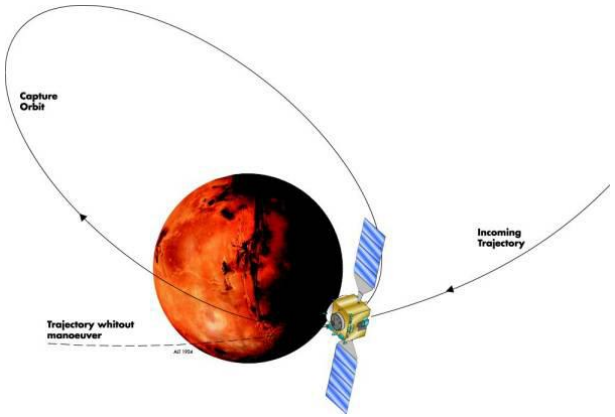
- Find:
  - A **sequence of actions** that achieves the goal
  - A **method** that is capable of finding a solution **for every application domain**

Use of planning in real applications

- Scheduling
- Hubble Space Telescope







Use of planning in real applications

- KIVA Robots: robots make inventory isles move instead of workers

youtube link

Use of planning in real applications

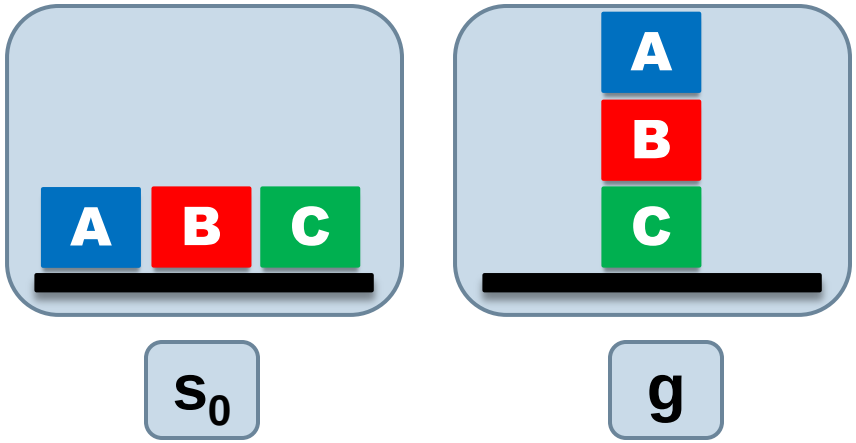
- Autonomous agents with planning capabilities for specifying the behavior (Intelligent Agents/ Cognitive Robots)
- UAV DARPA Grand/Urban Challenge, Honda ASIMO

STRIPS planning

- Typical description of a planning problem:
  - Initial state
  - Goal
  - Available actions

STRIPS planning

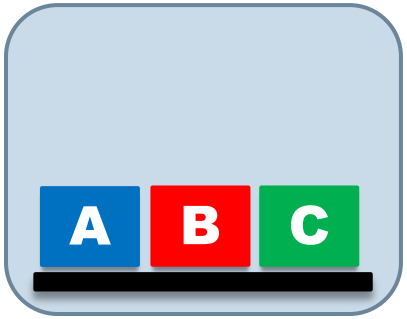
- Blocks world domain
  - Initial state: s_0
  - Goal: g



  - 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 block

STRIPS planning

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



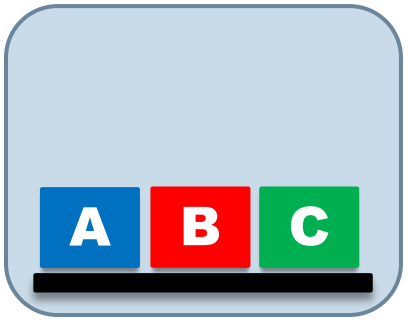
On(A,Table)
On(B,Table)
On(C,Table)
Clear(A)
Clear(B)
Clear(C)

- Blocks world
  - 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

STRIPS planning

- 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**



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

s_0

STRIPS planning

- 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)

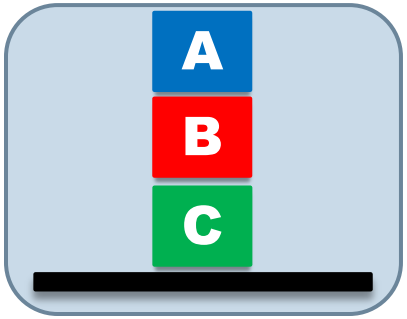
On(A,Table)
On(B,Table)
On(C,Table)
Clear(A)
Clear(B)
Clear(C)

S_0

STRIPS planning

- 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**)



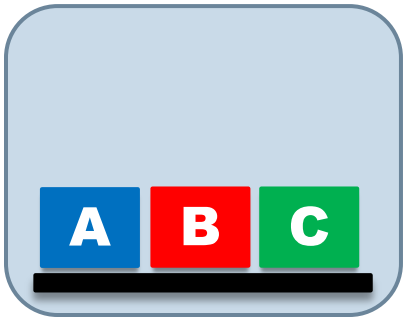
On(A,B)
On(B,C)

g

STRIPS planning

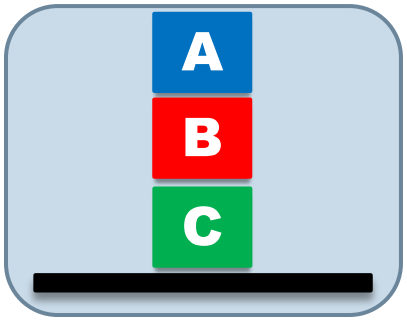
- The initial state and goal condition are described using first-order logic literals:
  - ground
  - function-free
  - positive

*(the list of literals typically forms a logical conjunction)



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

s_0



On(A,B)
On(B,C)

g

STRIPS planning

- 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

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

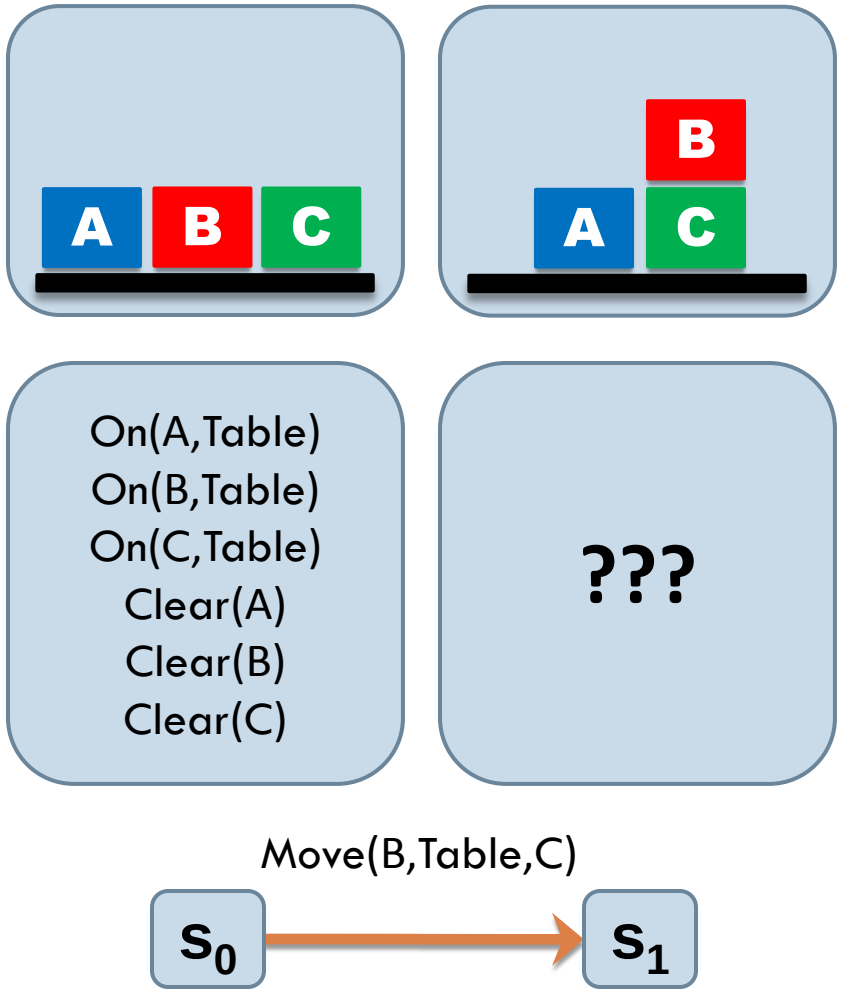
s_0

STRIPS planning

- 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



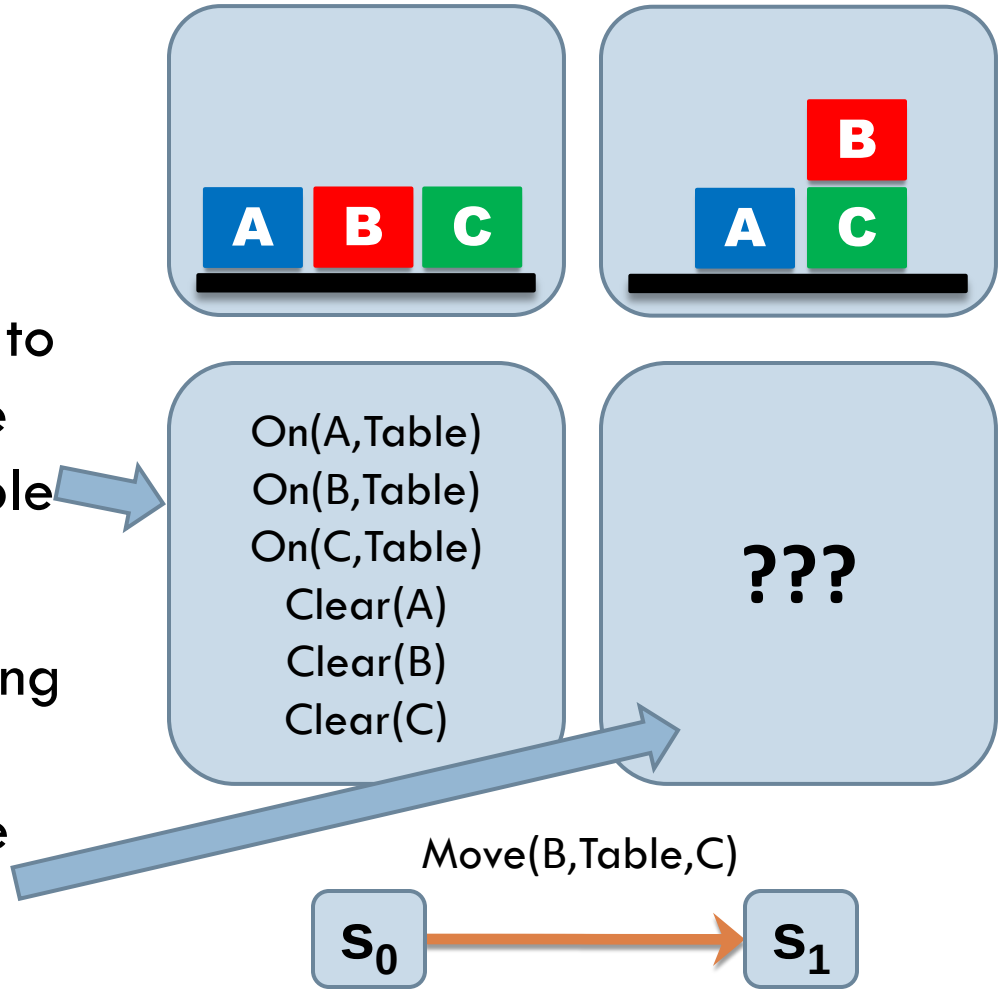
STRIPS planning

- 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.

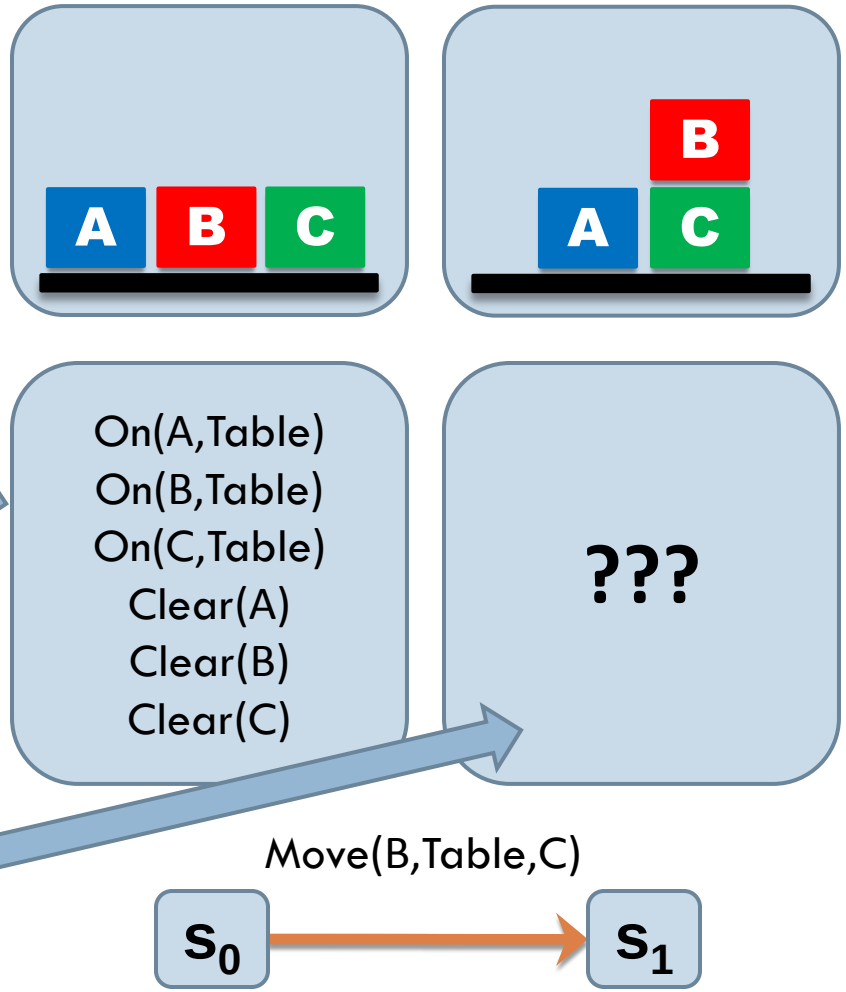


STRIPS planning

- 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)

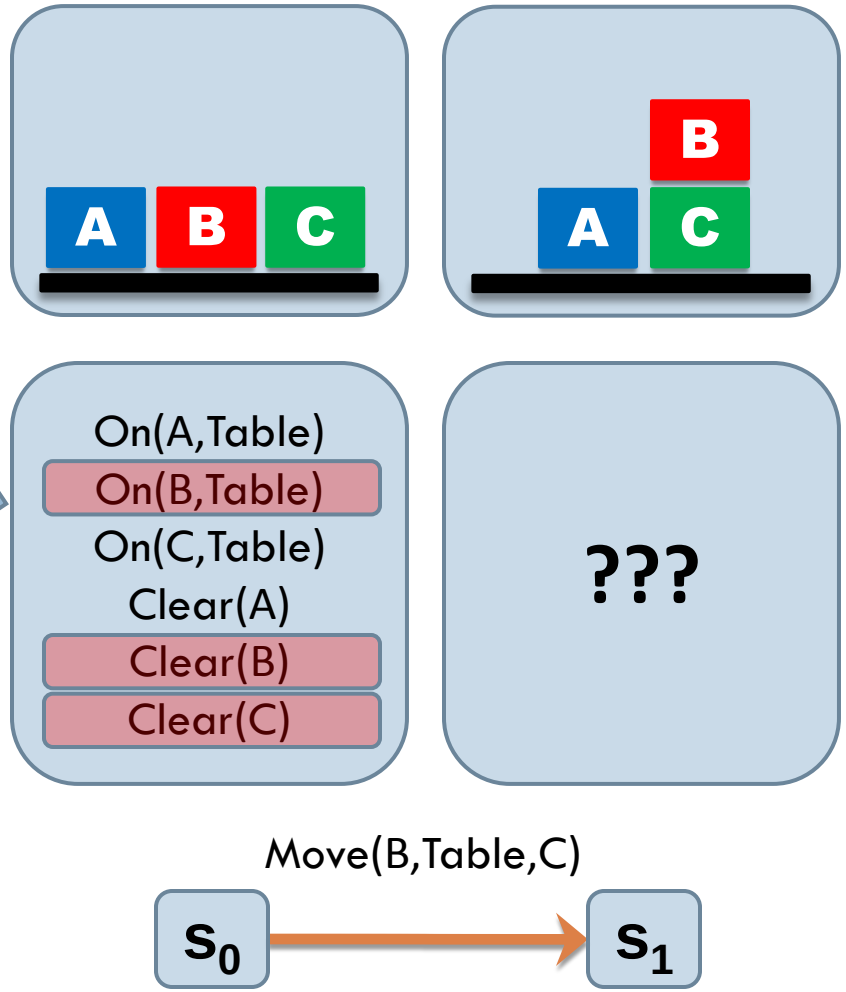


STRIPS planning

- Available actions

Move(B,Table,C)

- Preconditions:
  - **OK!** On(B,Table)
  - **OK!** Clear(B)
  - **OK!** Clear(C)
- Effects:
  - On(B,C)
  - Clear(Table)
  - ¬ On(B,Table)
  - ¬Clear(C)

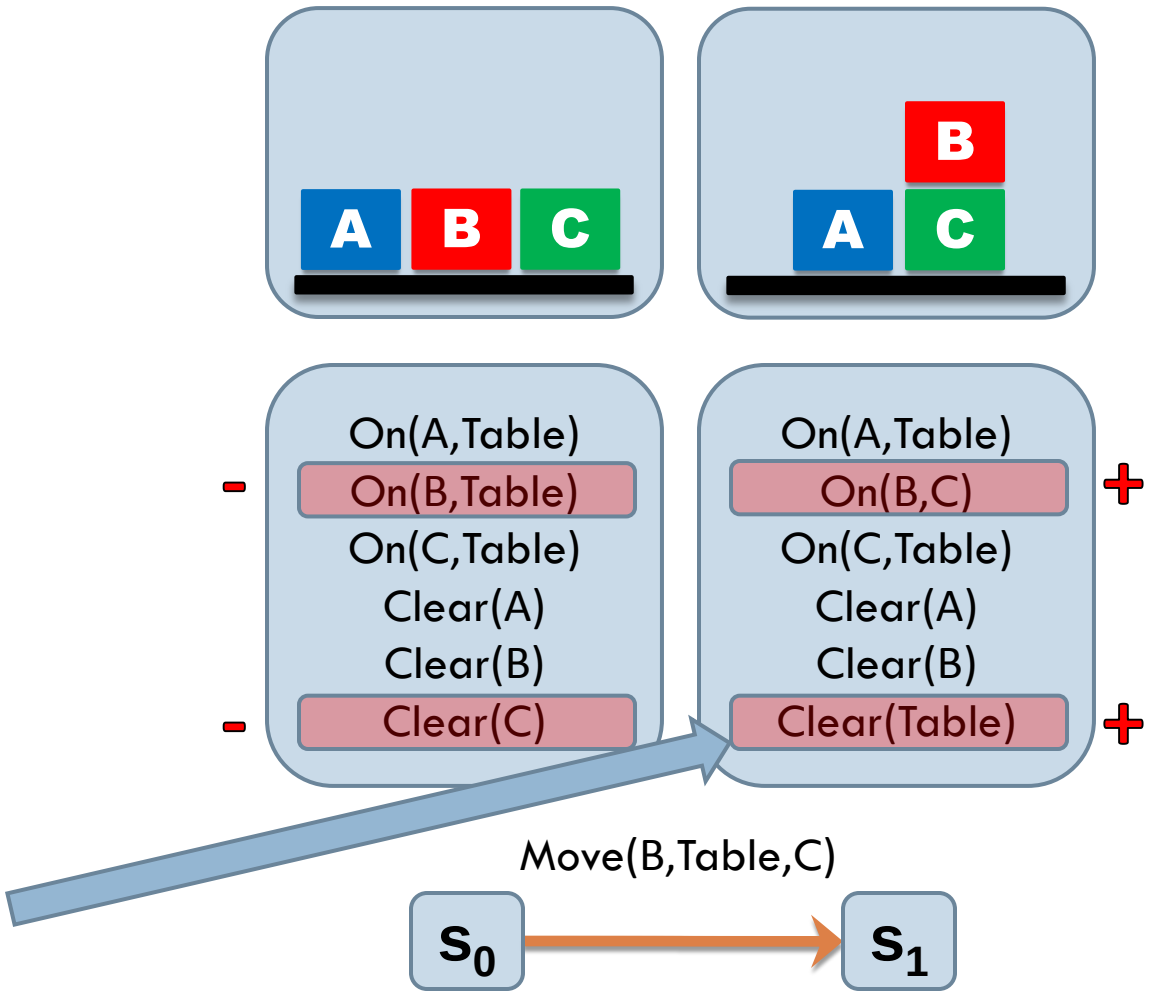


STRIPS planning

- Available actions

Move(B,Table,C)

- Preconditions:
  - On(B,Table)
  - Clear(B)
  - Clear(C)
- Effects:
  - \+ On(B,C)
  - \+ Clear(Table)
  - \- ¬ On(B,Table)
  - \- ¬Clear(C)

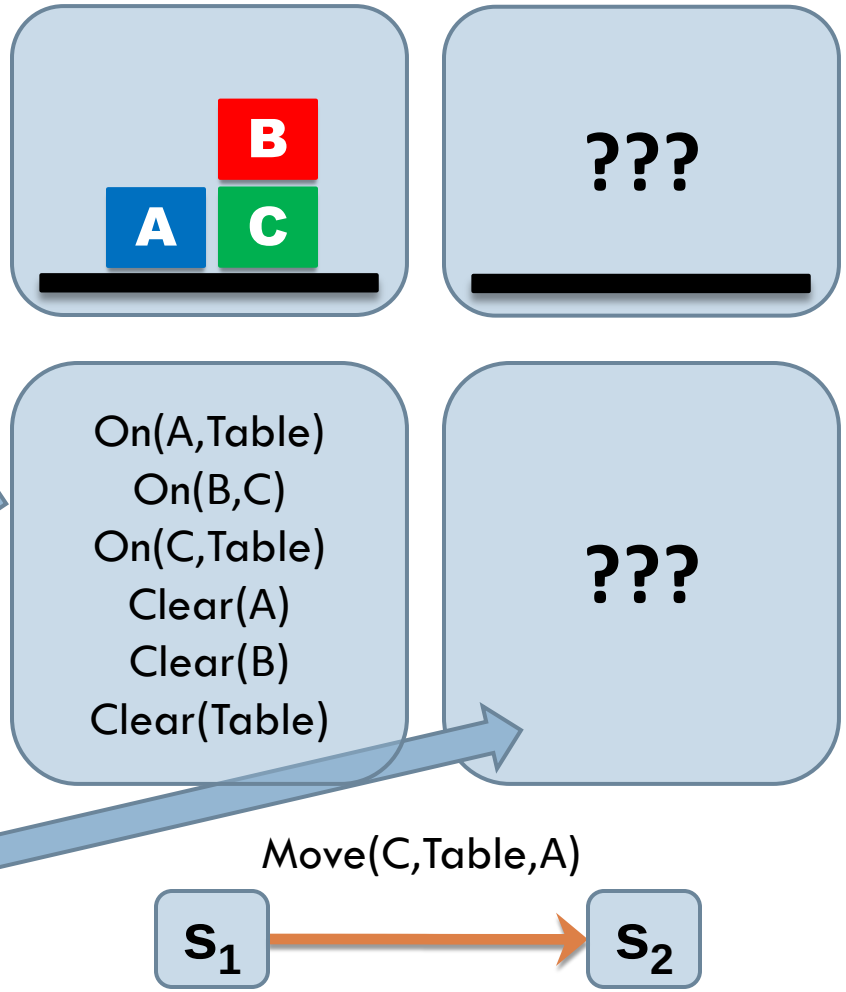


STRIPS planning

- 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)

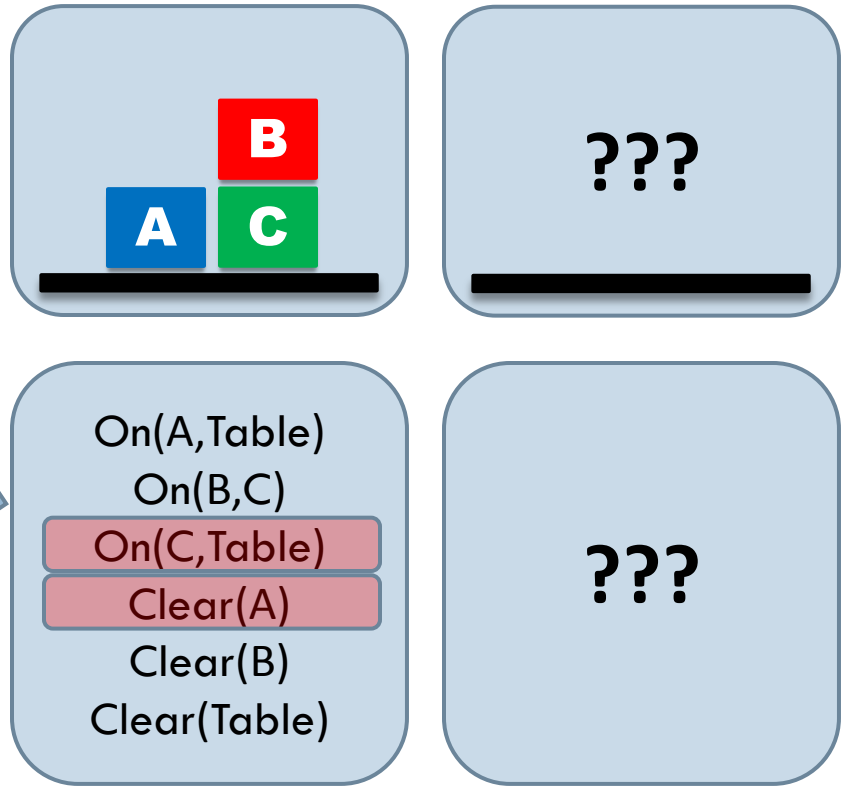


STRIPS planning

- Available actions

Move(C,Table,A)

- Preconditions:
  - **OK!** On(C,Table)
  - **X** Clear(C)
  - **OK!** Clear(A)
- Effects:
  - On(C,A)
  - Clear(Table)
  - ¬ On(C,Table)
  - ¬Clear(A)

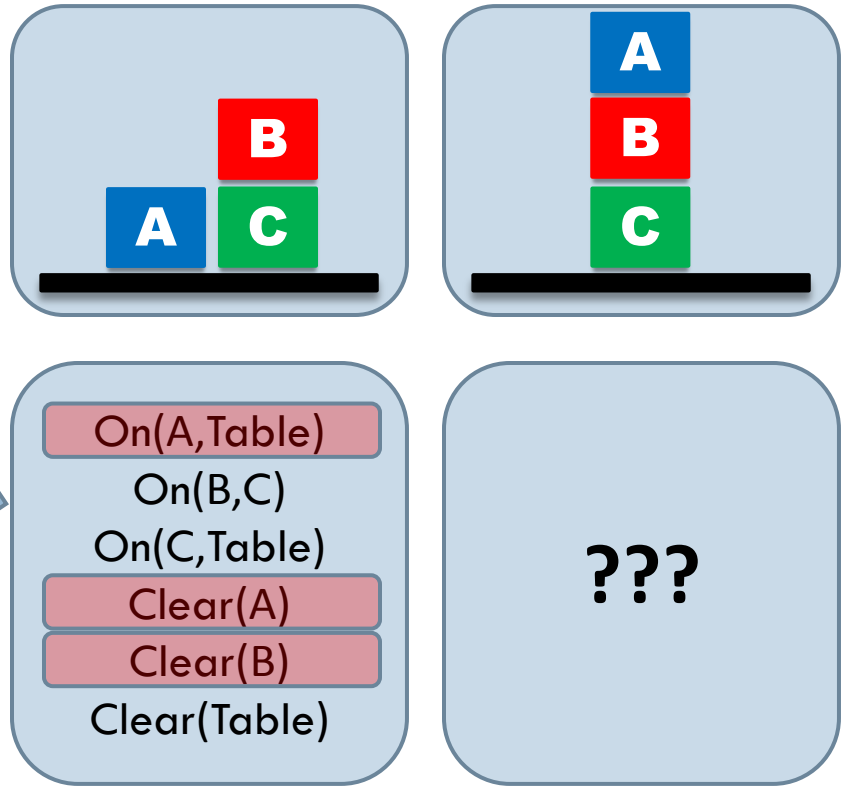


STRIPS planning

- Available actions

Move(A,Table,B)

- Preconditions:
  - OK! On(A,Table)
  - OK! Clear(A)
  - OK! Clear(B)
- Effects:
  - On(A,B)
  - Clear(Table)
  - ¬ On(A,Table)
  - ¬Clear(B)

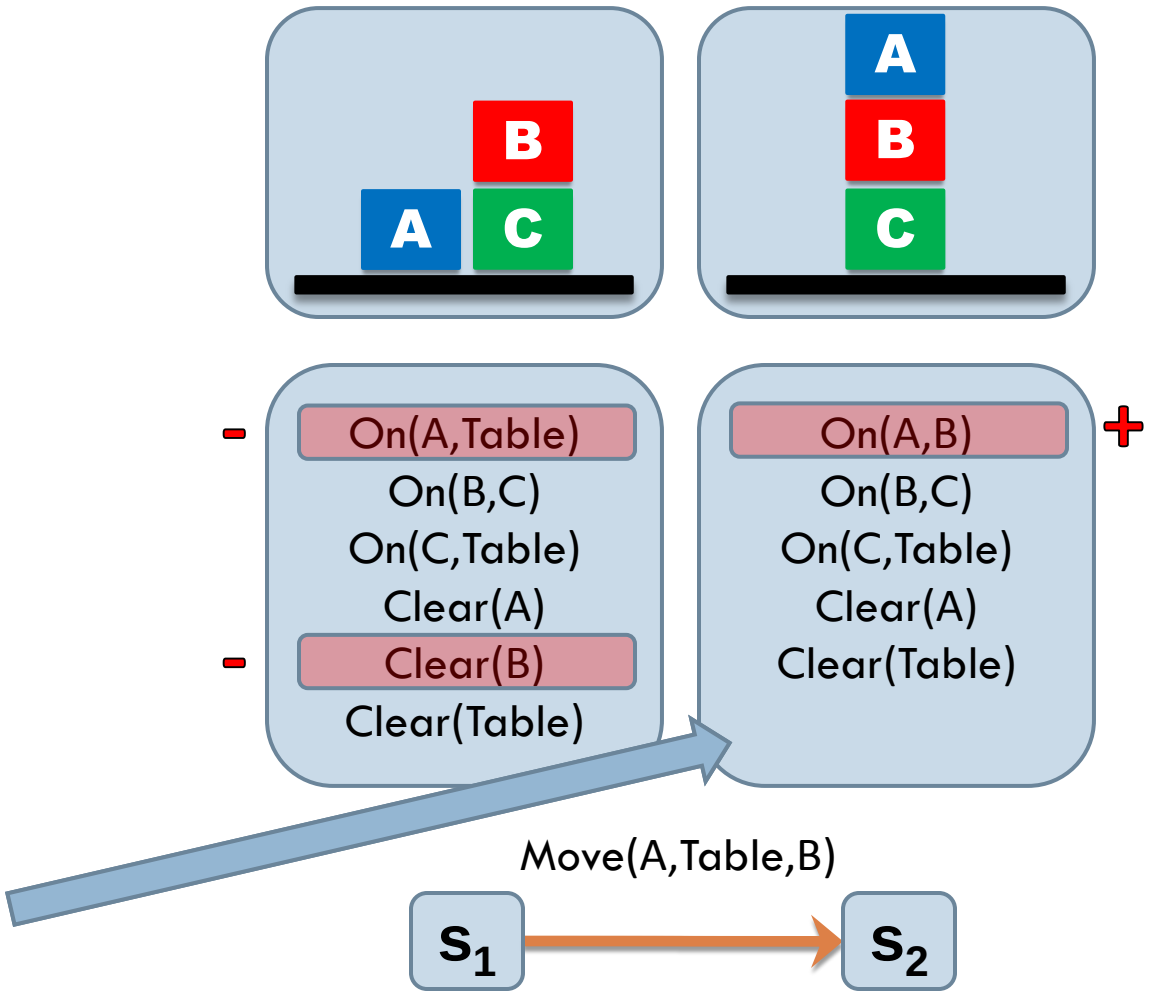


STRIPS planning

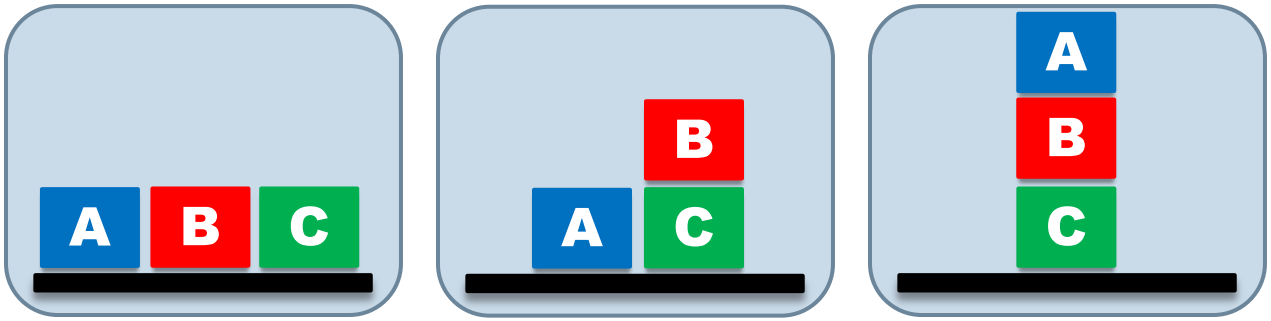
- Available actions

Move(A,Table,B)

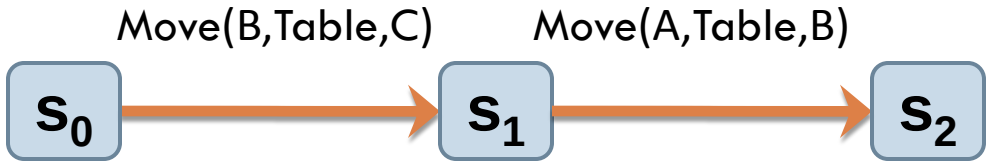
- Preconditions:
  - On(A,Table)
  - Clear(A)
  - Clear(B)
- Effects:
  - \+ On(A,B)
  - \+ Clear(Table)
  - \- ¬ On(A,Table)
  - \- ¬Clear(B)



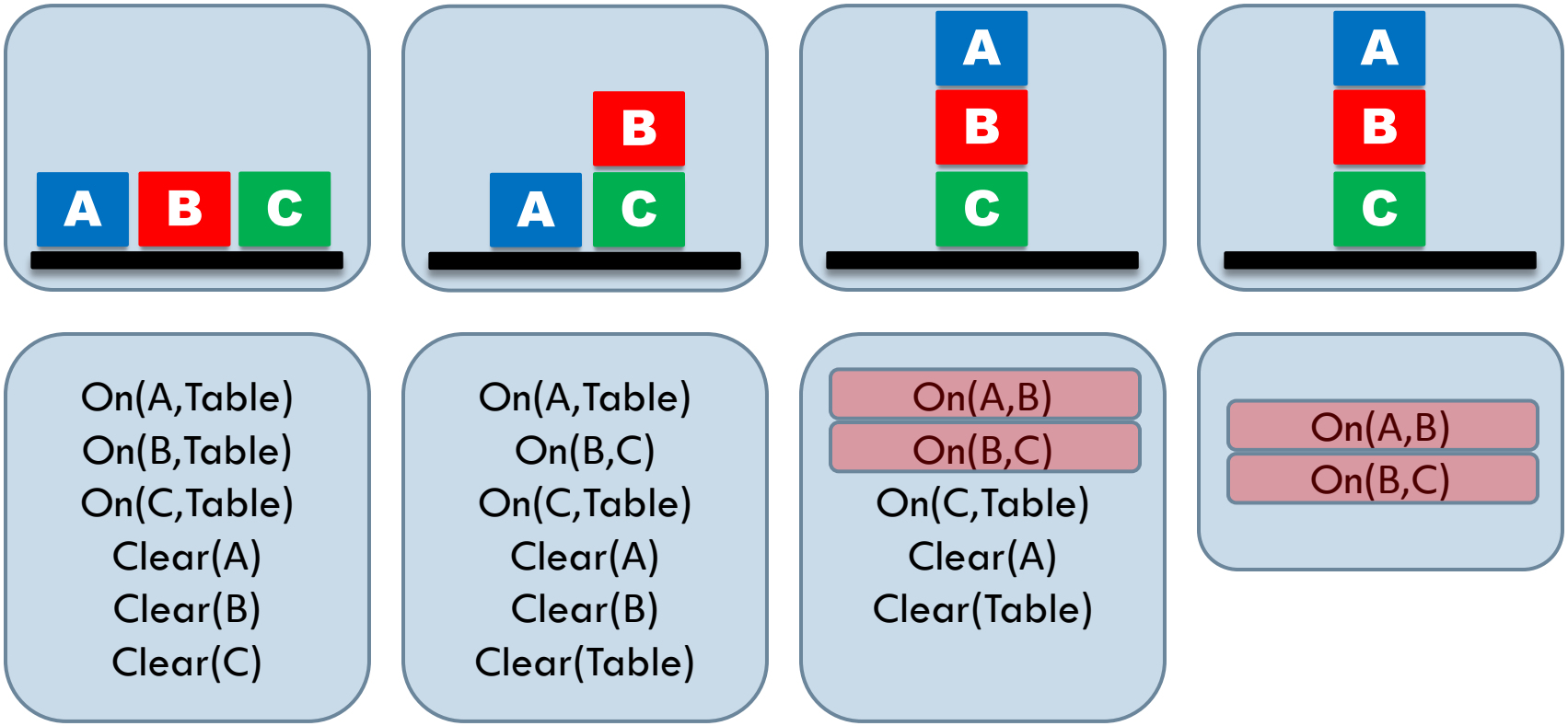
STRIPS planning



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

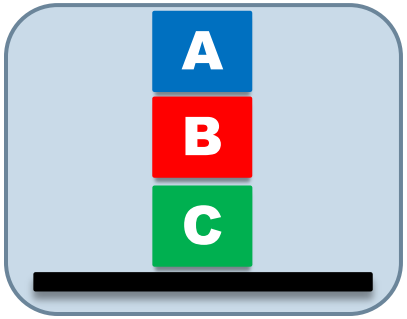


STRIPS planning



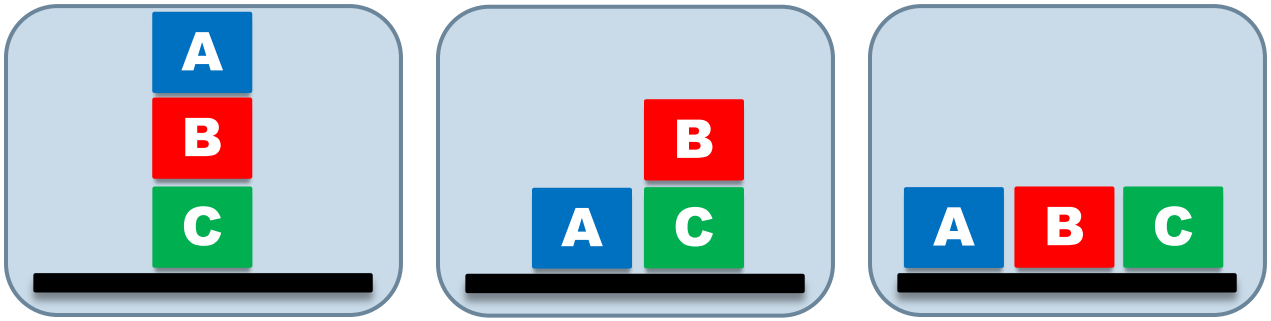
STRIPS planning

- Is our STRIPS representation of the blocks world correct?
- Consider moving back blocks A and B back to the table



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

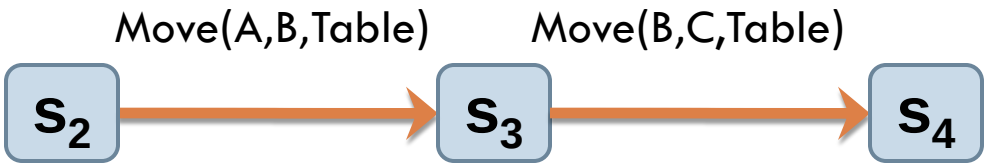
STRIPS planning



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

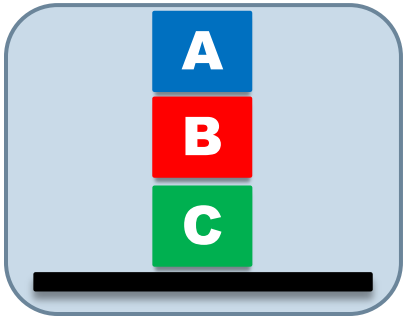
On(A,Table)
On(B,C)
On(C,Table)
Clear(A)
Clear(B)

???



STRIPS planning

- Is our STRIPS representation of the blocks world correct?
- Consider moving back blocks A and B back to the table



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

- Clear(Table) needs to be treated differently
  - Move(b,x,y)
  - MoveToTable(b,x)

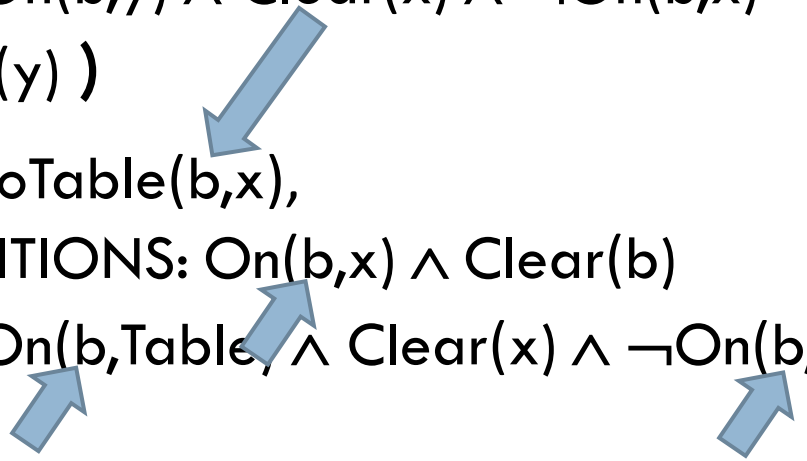
STRIPS planning

- 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))

STRIPS planning

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

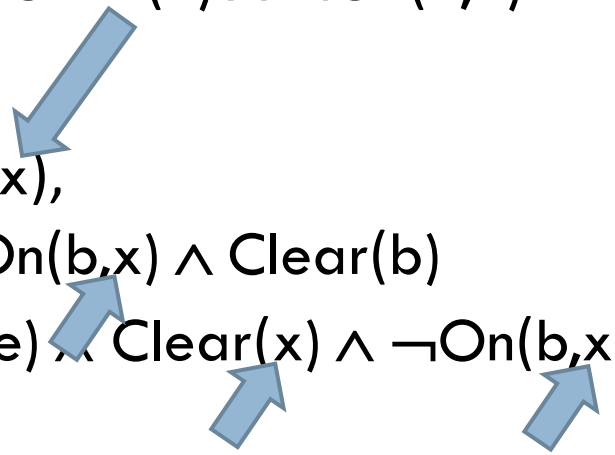
- Action(Move(b,x,y),
  PRECONDITIONS: On(b,x) ∧ Clear(b) ∧ Clear(y)
  EFFECTS: On(b,y) ∧ Clear(x) ∧ ¬On(b,x)
  ∧ ¬Clear(y))
- Action(MoveToTable(b,x),
  PRECONDITIONS: On(b,x) ∧ Clear(b)
  EFFECTS: On(b,Table) ∧ Clear(x) ∧ ¬On(b,x))



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))



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))

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))

! Simplified STRIPS version of Figure 11.4 in AIMA 2nd Ed.

STRIPS planning

- Gripper domain
  - Init(On-Table(A) ∧ On-Table(B) ∧ On-table(C)
    ∧ Clear(A) ∧ Clear(B) ∧ Clear(C) ∧ Grip-Empty)
  - Goal(On(A,B) ∧ On(B,C))
  - Action(Pick-up(x), …)
  - Action(Put-down(x), …)
  - Action(Stack(top,below), …)
  - Action(Unstack(top,below), …)

STRIPS planning

- Reasoning about action and change

- Since 1971 that STRIPS planning was proposed by R. E. Fikes and N. J. Nilsson, more expressive planning formalisms have been developed based on STRIPS!
  - ADL allows open worlds, conditional effects, quantifiers [Pednault 1988]

- Situation calculus [McCarthy, Hayes 1969] [Reiter 2001] is based on first-order logic allowing rich representations
  - Action languages A, Fluent calculus, Event calculus, ...

Bibliography

- Material
  - Artificial Intelligence: A Modern Approach 2nd Ed. Stuart Russell, Peter Norvig. Prentice Hall, 2003 Section 11.1
- References
  - STRIPS: A new approach to the application of theorem proving to problem solving. Richard E. Fikes, Nils J. Nilsson. Artificial Intelligence, Vol. 2, No. 3-4, 1971.
  - Synthesizing plans that contain actions with context-dependent effects. E. P. D. Pednault. Computational Intelligence, Vol. 4, No. 3. 1988.
  - Some Philosophical Problems from the Standpoint of Artificial Intelligence. John McCarthy, Patrick J. Hayes. Machine Intelligence, Vol. 4, 1969.
  - Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems. Raymond Reiter. MIT Press, 2001.