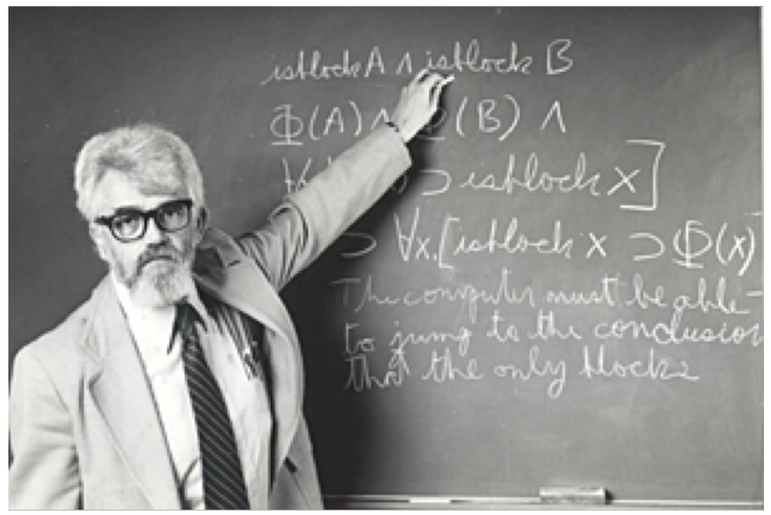
Classical Planning

CS 6300
Artificial Intelligence
Spring 2018
Tucker Hermans
thermans@cs.utah.edu
Presented by Rush Sanghrajka

Many slides from
Mike Stilman

Starting from the Beginning…of AI



Use a Model of the World to answer:

– What will happen next in a certain aspect of the situation?

– What will happen if I do a certain action?

– What is 3+3?

– What does he want?

– Can I figure out how to do this or must I get information from someone else or something else?

Predicate Logic (First-Order Logic)

World can be described by:

- Objects: *apple*, *myChair*, *house*, *mike*, *robot*, *suitcase*
- Variables: $x, y, z, \ldots$
- Relations (Predicates): *ParentOf(. , .), HasA(.), Smells(.)*
- Connectives: $\neg, \vee, \wedge, \Rightarrow$
- Functions: *Color(.), Height(.), Smell(.), GraspLoc(. , .)*

Other than functions, expressions evaluate to *true* or *false*

$$LessThan(1, 2)$$

$$\neg Contains(Suitcase, Bomb)$$

Planning: Situation Calculus

- **Fluents**: add state (or time) variable to relations:

  Contains(*Suitcase*, *Laptop*, **S0**)

  Working(*Robot*, *t*=4)

- **Actions:** are *reified* functions of constants (Can be treated as constants themselves)

  Put(Laptop, Suitcase) *Open(Car)* *Lock(Car)*

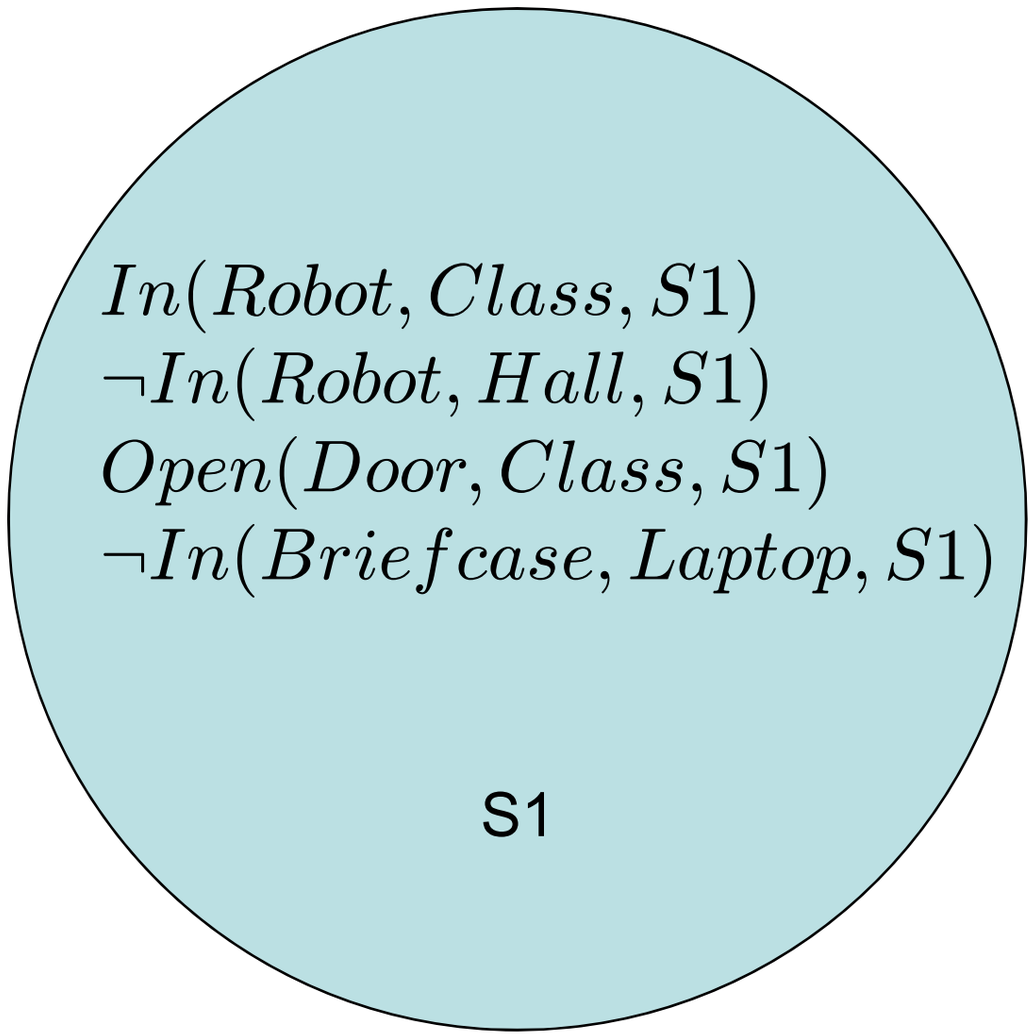
- The **do function**: $do(\alpha, \sigma_0) \rightarrow \sigma_1$

$$\alpha = action$$

$$\sigma = state$$

Situation Calculus: States

How should we describe a state?



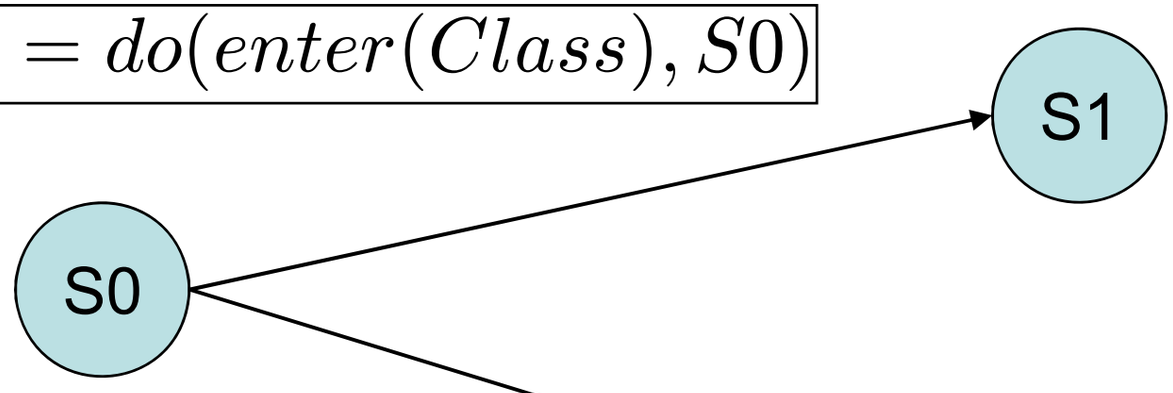
Situation Calculus: Actions

How about actions?

Effect axioms (Positive + Negative): $Room(rm)$

$$In(Robot, Hall, s) \wedge Open(rm, s) \Rightarrow In(Robot, rm, \mathbf{do}(\mathbf{enter}(\mathbf{rm}), \mathbf{s}))$$
$$In(Robot, Hall, s) \wedge Open(rm, s) \Rightarrow \neg In(Robot, Hall, \mathbf{do}(\mathbf{enter}(\mathbf{rm}), \mathbf{s}))$$

$\boxed{S1 = do(enter(Class), S0)}$



$In(Robot, Hall, S0)$
$Open(Class, S0)$
$Open(Office, S0)$

$In(Robot, Class, S1)$
$\neg In(Robot, Hall, S1)$

$In(Robot, Office, S2)$
$\neg In(Robot, Hall, S2)$

$\boxed{S2 = do(enter(Office), S0)}$

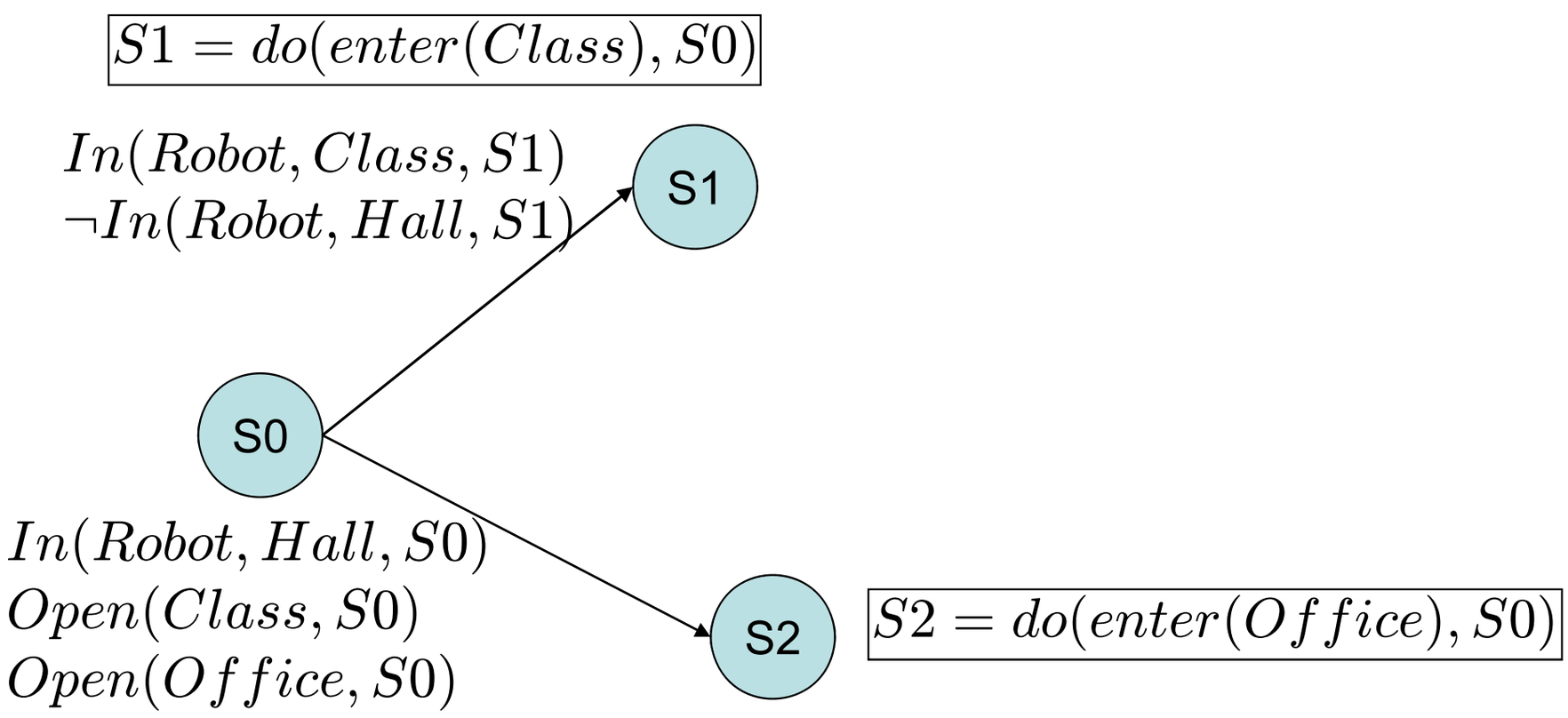
What else for planning?

Goals!

Use quantifiers!

Now what?

Plan!

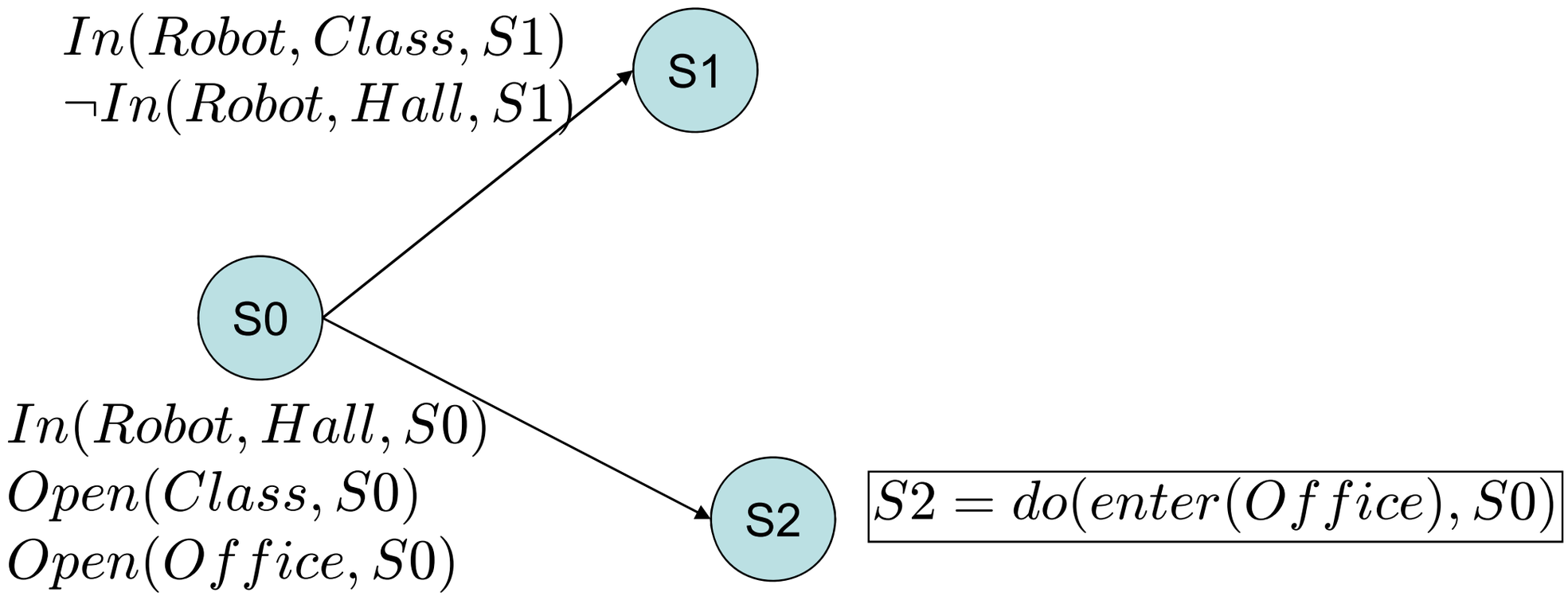


Goal: $\exists s Educated(Students, s)$

Situation Calculus: A Plan

$$In(Robot, rm, s) \Rightarrow Educated(Students, rm, \mathbf{do}(\mathbf{teach}(\mathbf{rm}), \mathbf{s}))$$

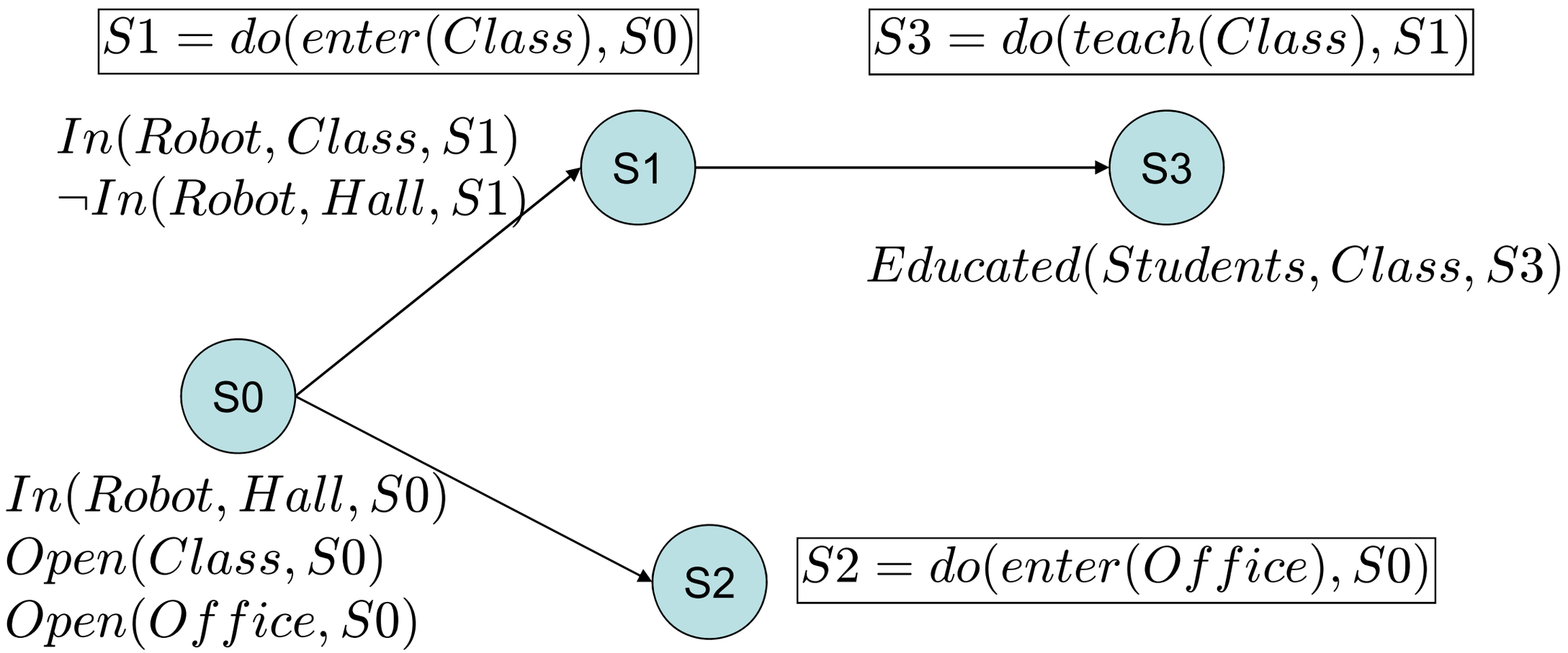
$\boxed{S1 = do(enter(Class), S0)}$



Goal: $\exists s Educated(Students, s)$

Situation Calculus: A Plan

$$In(Robot, rm, s) \Rightarrow Educated(Students, rm, \mathbf{do}(\mathbf{teach}(\mathbf{rm}), \mathbf{s}))$$



$$S3 = \mathbf{do}(teach, Class, \mathbf{do}(enter, Class, S0))$$

$$Educated(Students, S3)$$

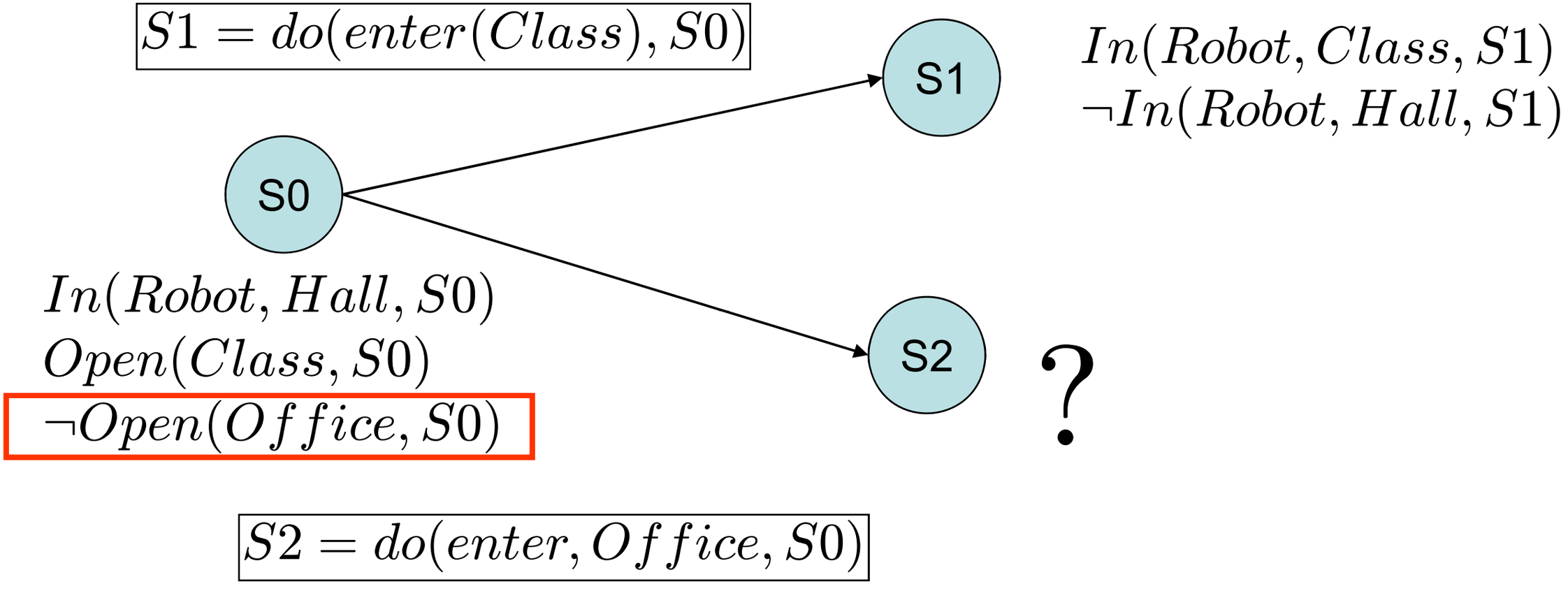
Situation Calculus: Problems?

$$Room(rm)$$

- Effect Axioms (Positive + Negative)

$$In(Robot, Hall, s) \wedge Open(rm, s) \Rightarrow In(Robot, rm, \mathbf{do}(\mathbf{enter}(\mathbf{rm}), \mathbf{s}))$$

$$In(Robot, Hall, s) \wedge Open(rm, s) \Rightarrow \neg In(Robot, Hall, \mathbf{do}(\mathbf{enter}(\mathbf{rm}), \mathbf{s}))$$

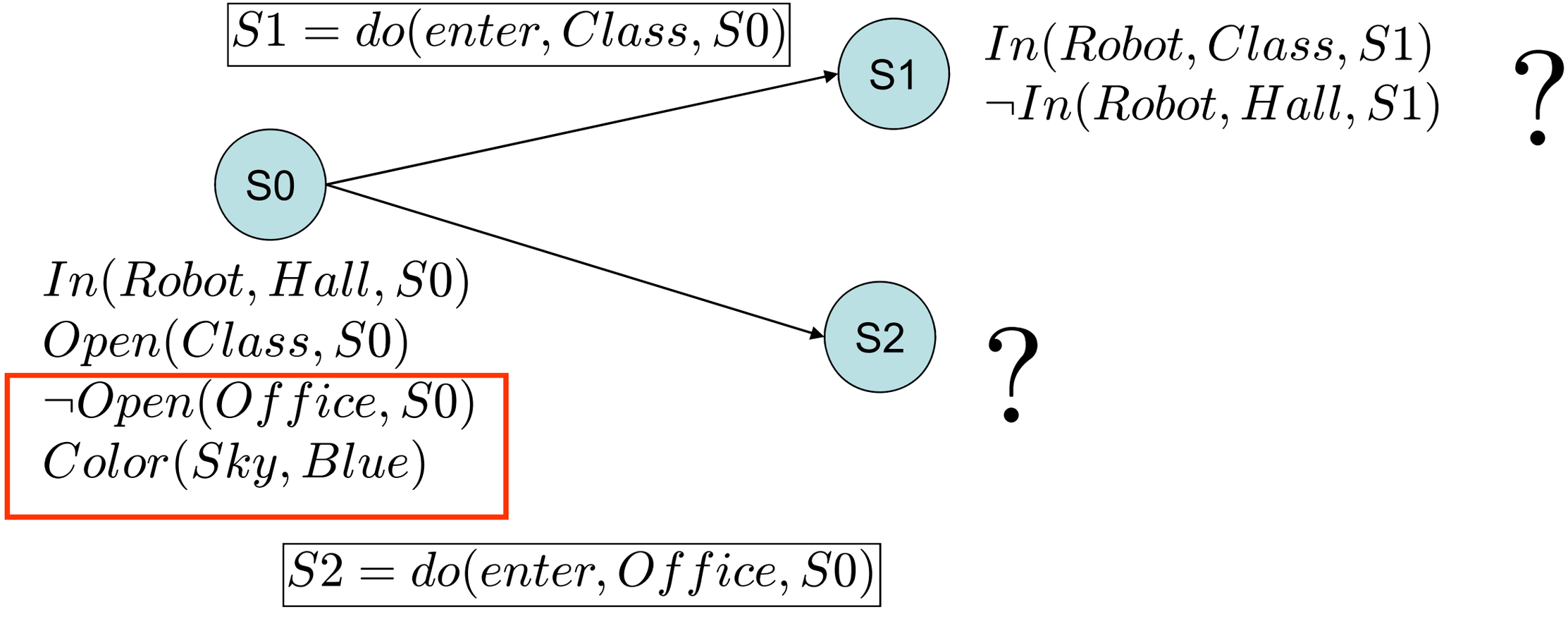


Situation Calculus: Frame Problem

$$Room(rm)$$

- Effect Axioms (Positive + Negative)

$$In(Robot, Hall, s) \wedge Open(rm, s) \Rightarrow In(Robot, rm, \mathbf{do}(\mathbf{enter}(\mathbf{rm}), \mathbf{s}))$$
$$In(Robot, Hall, s) \wedge Open(rm, s) \Rightarrow \neg In(Robot, Hall, \mathbf{do}(\mathbf{enter}(\mathbf{rm}), \mathbf{s}))$$



Frame Axioms

Effect Axioms (Positive + Negative)

$$In(Robot, Hall, s) \wedge Open(rm, s) \Rightarrow In(Robot, rm, \mathbf{do}(\mathbf{enter}(\mathbf{rm}), \mathbf{s}))$$

$$In(Robot, Hall, s) \wedge Open(rm, s) \Rightarrow \neg In(Robot, Hall, \mathbf{do}(\mathbf{enter}(\mathbf{rm}), \mathbf{s}))$$

For each unchanged fluent we add:

$$Open(Office, s) \Rightarrow Open(Office, \mathbf{do}(\mathbf{enter}(\mathbf{rm}), \mathbf{s}))$$
$$\neg Open(Office, s) \Rightarrow \neg Open(Office, \mathbf{do}(\mathbf{enter}(\mathbf{rm}), \mathbf{s}))$$

$$Color(Sky, Blue, s) \Rightarrow Color(Sky, Blue, \mathbf{do}(\mathbf{enter}(\mathbf{rm}), \mathbf{s}))$$
$$\neg Color(Sky, Blue, s) \Rightarrow \neg Color(Sky, Blue, \mathbf{do}(\mathbf{enter}(\mathbf{rm}), \mathbf{s}))$$

How many in total? (for n distinct fluents and m distinct actions)

$$2nm$$ (Not exponential – but often not practical)

Ramification Problem

What are the ramifications of an action?

- The robot entered the room.
- Its sensors are now in the room.
- The object it was carrying is now in the room.
- The robot is visible to you.

Do we really want to say all that in an effect axiom?

How does this relate to the Frame Problem?

Classical Planning

- Finite number of states
- Fully observable
- Deterministic
- Static
- Discrete
- Solution Plan

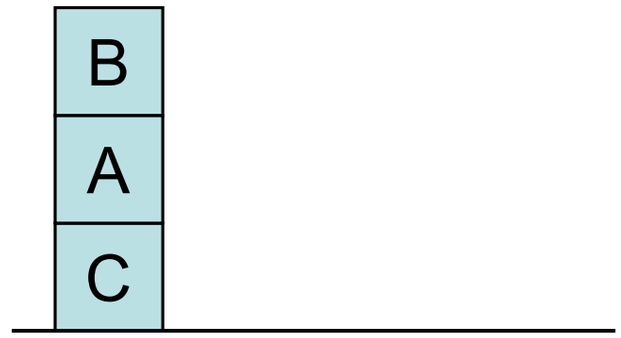
STRIPS (Fikes & Nilsson 1971)

- Restrict state descriptions to conjunctions of fluents (literals)
  - Why?
  - What do we gain? Lose?

- Represent actions (operators) as three parts:
  - PC: set of preconditions
  - D: Delete List
  - A: Add List

- Use recursive search to implement planning

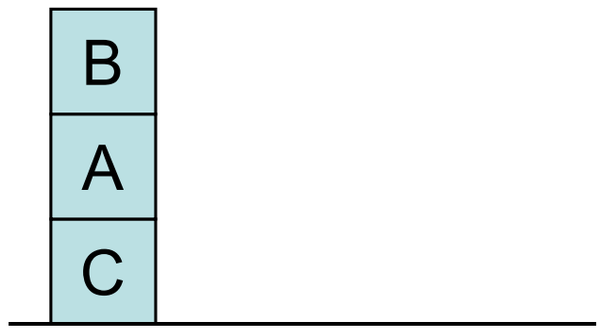
STRIPS: Blocks World



STRIPS: Blocks World

Constants:

$$A, B, C, Table$$
$$IsBlock(A), IsBlock(B)...$$



Ground Literals:

$$On(B, A) \quad On(C, Table)$$
$$Clear(B) \quad Clear(Table)$$

Actions: $move(x, y, z)$

PC: $On(x, y)$, $Clear(x)$, $Clear(z)$

D: $On(x, y)$, $Clear(z)$

A: $On(x, z)$, $Clear(y)$, $Clear(Table)$

Domain Axioms:

$$On(y, x) \wedge On(z, x) \wedge (x \neq Table) \Rightarrow (y = z)$$

STRIPS: Blocks World

Constants:

$$A, B, C, Table$$
$$IsBlock(A), IsBlock(B)...$$

Ground Literals:

$$On(B, A) \quad On(C, Table)$$
$$Clear(B) \quad Clear(Table)$$

Actions: $move(x, y, z)$

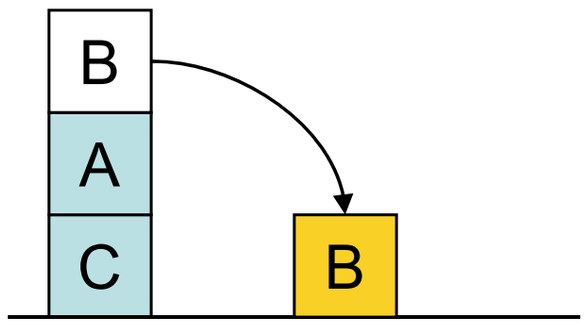
PC: $On(x, y)$ $Clear(x)$ $Clear(z)$

D: $On(x, y)$ $Clear(z)$

A: $On(x, z)$ $Clear(y)$ $Clear(Table)$

Domain Axioms:

$$On(y, x) \wedge On(z, x) \wedge (x \neq Table) \Rightarrow (y = z)$$



$move(B, A, Table)$

S0

Delete:

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

Unchanged

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

S1

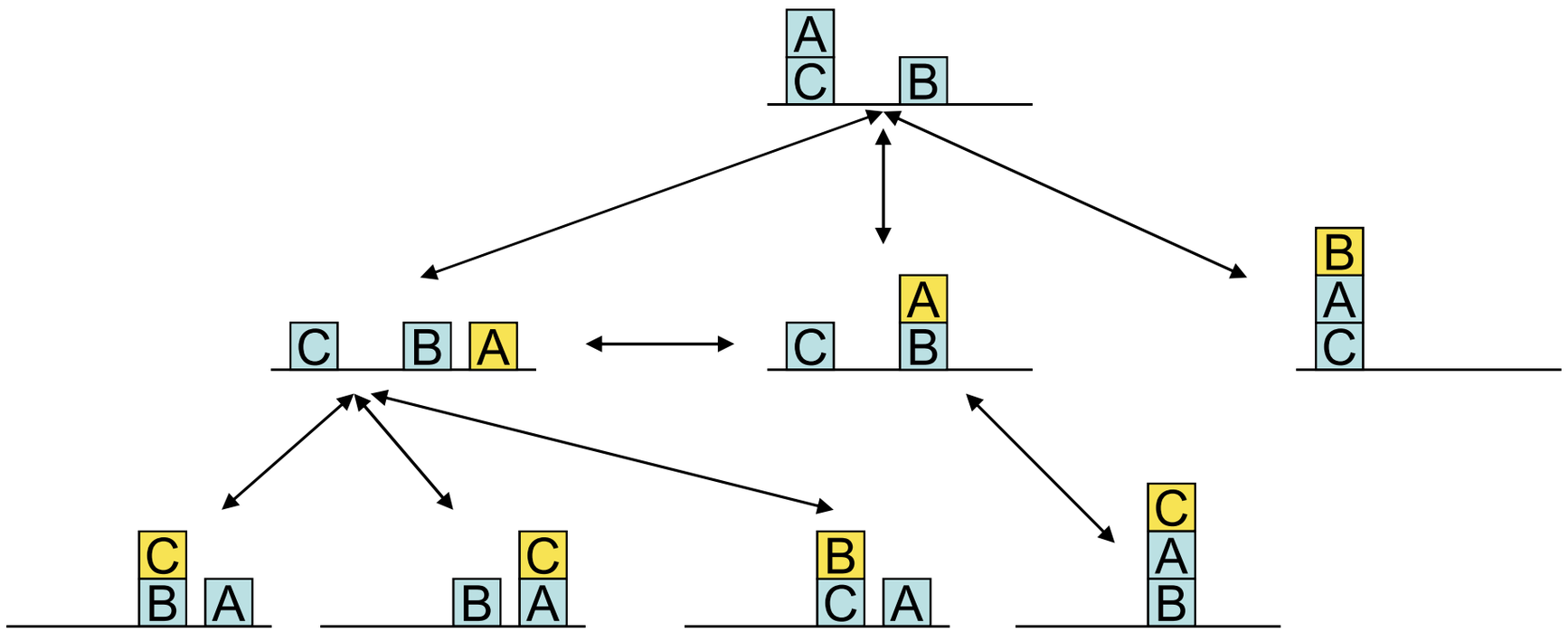
Add:

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

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

How to plan in strips?

Search!



Linear Search: How STRIPS works

- Desire to speed up search

- Non-linear planning:
  - Interleave search for subgoals
  - Keep a **set** of unachieved goals

- Linear planning:
  - Solve one goal at a time
  - Search a **stack** of unachieved goals

Based on General Problem Solver (GPS)

- Newell, Simon, & Ernst – 1960's
- Introduced *means-end analysis*
- Use a recursive goal stack to speed up planning

Fikes & Nilson 1971

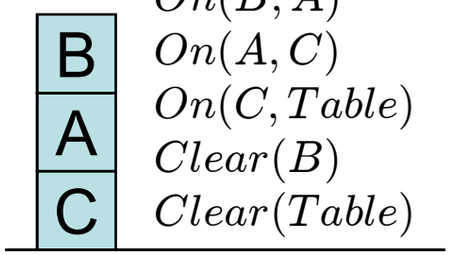
STRIPS(*init*,*goals*)

1. *state* = *init*, *plan* = [], *stack* = []
2. Push *goals* on *stack*
3. Repeat until *stack* = []
   - If *top* = *state* Pop *stack*
   - If *top* = conjunctive then Push subgoals onto *stack*.
   - If *top* = simple literal
     - Choose operator O where A_O = *top*
     - Replace *top* with operator O
     - Push preconditions of O onto *stack*
   - If top = operator O then
     - *state* = apply(O,*state*)
     - *plan* = [*plan*, O]

STRIPS: Solve Blocks World

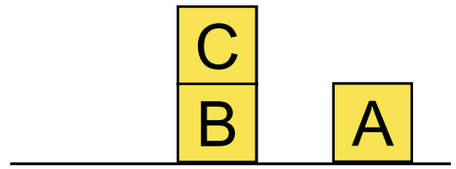
	Stack:	State:
1		$On(B,A)$ $On(A,C)$ $On(C,Table)$ $Clear(B)$ $Clear(Table)$
	$On(A,Table) \wedge On(B,Table) \wedge On(C,B)$	

init



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

goal



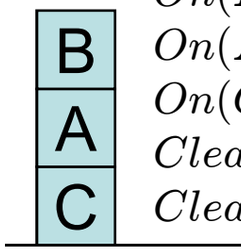
$On(A,Table) \wedge On(B,Table) \wedge On(C,B)$

STRIPS(*init*,*goals*)
1. *state* = *init*, *plan* = [], *stack* = []
2. Push *goals* on *stack*
3. Repeat until *stack* = []
   - If *top* = *state* Pop *stack*
   - If *top* = conjunctive then Push subgoals onto *stack*.
   - If *top* = simple literal
     - Choose operator O where A_O = *top*
     - Replace *top* with operator O
     - Push preconditions of O onto *stack*
   - If top = operator O then
     - *state* = apply(O,*state*)
     - *plan* = [*plan*, O]

STRIPS: Solve Blocks World

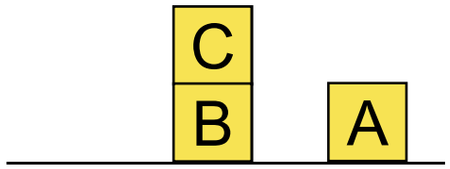
	Stack:	State:
1	$On(A,Table) \wedge On(B,Table) \wedge On(C,B)$	$On(B,A)$ $On(A,C)$ $On(C,Table)$ $Clear(B)$ $Clear(Table)$
2	$On(A,Table)$ $On(B,Table)$ $On(C,B)$ $On(A,Table) \wedge On(B,Table) \wedge On(C,B)$	$On(B,A)$ $On(A,C)$ $On(C,Table)$ $Clear(B)$ $Clear(Table)$

init



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

goal



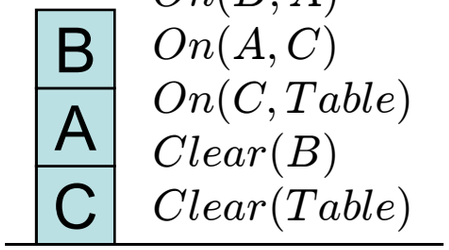
$On(A,Table) \wedge On(B,Table) \wedge On(C,B)$

STRIPS(*init*,*goals*)
1. *state* = *init*, *plan* = [], *stack* = []
2. Push *goals* on *stack*
3. Repeat until *stack* = []
   - If *top* = *state* Pop *stack*
   - If *top* = conjunctive then Push subgoals onto *stack*.
   - If *top* = simple literal
     - Choose operator O where A_O = *top*
     - Replace *top* with operator O
     - Push preconditions of O onto *stack*
   - If top = operator O then
     - *state* = apply(O,*state*)
     - *plan* = [*plan*, O]

STRIPS: Solve Blocks World

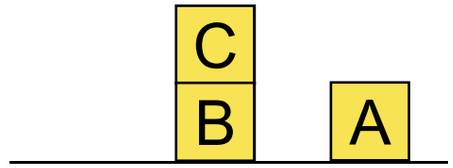
	Stack:	State:
1	$On(A,Table) \wedge On(B,Table) \wedge On(C,B)$	$On(B,A)$ $On(A,C)$ $On(C,Table)$ $Clear(B)$ $Clear(Table)$
2	$On(A,Table)$ $On(B,Table)$ $On(C,B)$ $On(A,Table) \wedge On(B,Table) \wedge On(C,B)$	$On(B,A)$ $On(A,C)$ $On(C,Table)$ $Clear(B)$ $Clear(Table)$
3	$On(A,C)$ $Clear(A)$ $Clear(Table)$ $move(A,C,Table)$ $On(B,Table)$ $On(C,B)$ $On(A,Table) \wedge On(B,Table) \wedge On(C,B)$	$On(B,A)$ $On(A,C)$ $On(C,Table)$ $Clear(B)$ $Clear(Table)$

init



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

goal



$On(A,Table) \wedge On(B,Table) \wedge On(C,B)$

	PC	D	A
$move(x,y,z)$	$On(x,y)$ $Clear(x)$ $Clear(z)$	$On(x,y)$ $Clear(z)$	$On(x,z)$ $Clear(y)$ $Clear(Table)$

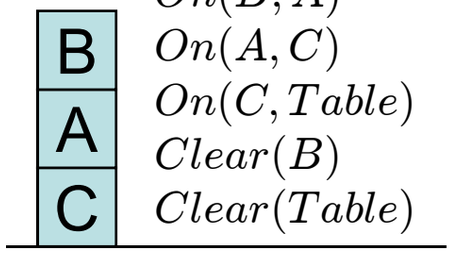
STRIPS(*init*,*goals*)

1. *state* = *init*, *plan* = [], *stack* = []
2. Push *goals* on *stack*
3. Repeat until *stack* = []
   - If *top* = *state* Pop *stack*
   - If *top* = conjunctive then Push subgoals onto *stack*.
   - If *top* = simple literal
     - Choose operator O where A_O = *top*
     - Replace *top* with operator O
     - Push preconditions of O onto *stack*
   - If top = operator O then
     - *state* = apply(O,*state*)
     - *plan* = [*plan*, O]

STRIPS: Solve Blocks World

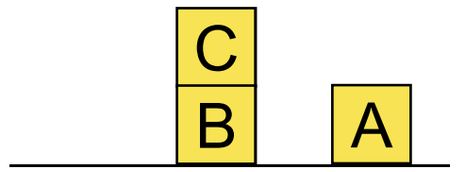
	Stack:	State:
1	$On(A,Table) \wedge On(B,Table) \wedge On(C,B)$	$On(B,A)$ $On(A,C)$ $On(C,Table)$ $Clear(B)$ $Clear(Table)$
2	$On(A,Table)$ $On(B,Table)$ $On(C,B)$ $On(A,Table) \wedge On(B,Table) \wedge On(C,B)$	$On(B,A)$ $On(A,C)$ $On(C,Table)$ $Clear(B)$ $Clear(Table)$
3	$On(A,C)$ $Clear(A)$ $Clear(C)$ $move(A,C,Table)$ $On(B,Table)$ $On(C,B)$ $On(A,Table) \wedge On(B,Table) \wedge On(C,B)$	$On(B,A)$ $On(A,C)$ $On(C,Table)$ $Clear(B)$ $Clear(Table)$
4	$Clear(A)$ $Clear(C)$ $move(A,C,Table)$ $On(B,Table)$ $On(C,B)$ $On(A,Table) \wedge On(B,Table) \wedge On(C,B)$	$On(B,A)$ $On(A,C)$ $On(C,Table)$ $Clear(B)$ $Clear(Table)$

init



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

goal



$On(A,Table) \wedge On(B,Table) \wedge On(C,B)$

	PC	D	A
$move(x,y,z)$	$On(x,y)$ $Clear(x)$ $Clear(z)$	$On(x,y)$ $Clear(z)$	$On(x,z)$ $Clear(y)$ $Clear(Table)$

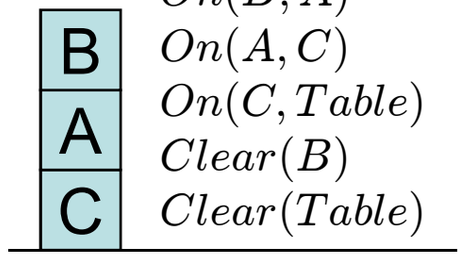
STRIPS(*init*,*goals*)

1. *state* = *init*, *plan* = [], *stack* = []
2. Push *goals* on *stack*
3. Repeat until *stack* = []
   - If *top* = *state* Pop *stack*
   - If *top* = conjunctive then Push subgoals onto *stack*.
   - If *top* = simple literal
     - Choose operator O where A_O = *top*
     - Replace *top* with operator O
     - Push preconditions of O onto *stack*
   - If top = operator O then
     - *state* = apply(O,*state*)
     - *plan* = [*plan*, O]

STRIPS: Solve Blocks World

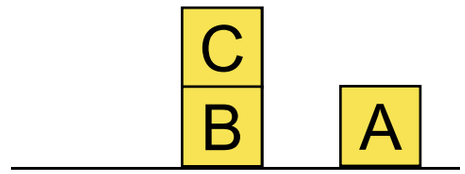
	Stack:	State:
4	$Clear(A)$	$On(B,A)$
	$Clear(Table)$	$On(A,C)$
	$move(A,C,Table)$	$On(C,Table)$
	$On(B,Table)$	$Clear(B)$
	$On(C,B)$	$Clear(Table)$
	$On(A,Table) \wedge On(B,Table) \wedge On(C,B)$	
5	$On(B,A)$	$On(B,A)$
	$Clear(B)$	$On(A,C)$
	$Clear(Table)$	$On(C,Table)$
	$move(B,A,Table)$	$Clear(B)$
	$Clear(Table)$	$Clear(Table)$
	$move(A,C,Table)$	
	$On(B,Table)$	
	$On(C,B)$	
	$On(A,Table) \wedge On(B,Table) \wedge On(C,B)$	

init



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

goal



$On(A,Table) \wedge On(B,Table) \wedge On(C,B)$

	PC	D	A
$move(x,y,z)$	$On(x,y)$ $Clear(x)$ $Clear(z)$	$On(x,y)$ $Clear(z)$	$On(x,z)$ $Clear(y)$ $Clear(Table)$

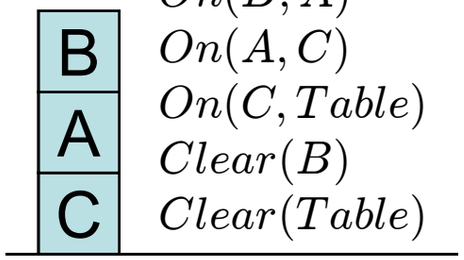
STRIPS(*init*,*goals*)

1. *state* = *init*, *plan* = [], *stack* = []
2. Push *goals* on *stack*
3. Repeat until *stack* = []
   - If *top* = *state* Pop *stack*
   - If *top* = conjunctive then Push subgoals onto *stack*.
   - If *top* = simple literal
     - Choose operator O where A_O = *top*
     - Replace *top* with operator O
     - Push preconditions of O onto *stack*
   - If top = operator O then
     - *state* = apply(O,*state*)
     - *plan* = [*plan*, O]

STRIPS: Solve Blocks World

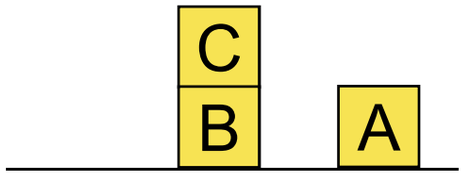
	Stack:	State:
4	$Clear(A)$ $Clear(Table)$ $move(A, C, Table)$ $On(B, Table)$ $On(C, B)$ $On(A, Table) \wedge On(B, Table) \wedge On(C, B)$	$On(B, A)$ $On(A, C)$ $On(C, Table)$ $Clear(B)$ $Clear(Table)$
5	$On(B, A)$ $Clear(B)$ $Clear(Table)$ $move(B, A, Table)$ $Clear(Table)$ $move(A, C, Table)$ $On(B, Table)$ $On(C, B)$ $On(A, Table) \wedge On(B, Table) \wedge On(C, B)$	$On(B, A)$ $On(A, C)$ $On(C, Table)$ $Clear(B)$ $Clear(Table)$
6	$move(B, A, Table)$ $Clear(Table)$ $move(A, C, Table)$ $On(B, Table)$ $On(C, B)$ $On(A, Table) \wedge On(B, Table) \wedge On(C, B)$	$On(B, A)$ $On(A, C)$ $On(C, Table)$ $Clear(B)$ $Clear(Table)$

init



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

goal



$On(A, Table) \wedge On(B, Table) \wedge On(C, B)$

	PC	D	A
$move(x, y, z)$	$On(x, y)$ $Clear(x)$ $Clear(z)$	$On(x, y)$ $Clear(z)$	$On(x, z)$ $Clear(y)$ $Clear(Table)$

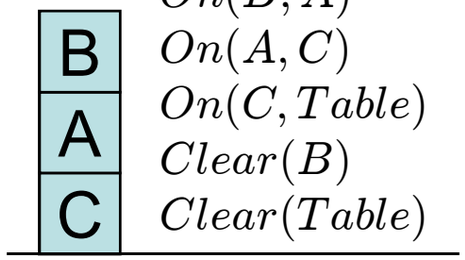
STRIPS(*init*,*goals*)

1. *state* = *init*, *plan* = [], *stack* = []
2. Push *goals* on *stack*
3. Repeat until *stack* = []
   - If *top* = *state* Pop *stack*
   - If *top* = conjunctive then Push subgoals onto *stack*.
   - If *top* = simple literal
     - Choose operator O where A_O = *top*
     - Replace *top* with operator O
     - Push preconditions of O onto *stack*
   - If top = operator O then Pop
     - *state* = apply(O,*state*)
     - *plan* = [*plan*, O]

STRIPS: Solve Blocks World

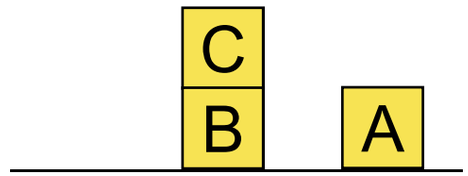
	Stack:	State:
4	$Clear(A)$ $Clear(Table)$ $move(A, C, Table)$ $On(B, Table)$ $On(C, B)$ $On(A, Table) \wedge On(B, Table) \wedge On(C, B)$	$On(B, A)$ $On(A, C)$ $On(C, Table)$ $Clear(B)$ $Clear(Table)$
5	$On(B, A)$ $Clear(B)$ $Clear(Table)$ $move(B, A, Table)$ $Clear(Table)$ $move(A, C, Table)$ $On(B, Table)$ $On(C, B)$ $On(A, Table) \wedge On(B, Table) \wedge On(C, B)$	$On(B, A)$ $On(A, C)$ $On(C, Table)$ $Clear(B)$ $Clear(Table)$
6	$move(B, A, Table)$ $Clear(Table)$ $move(A, C, Table)$ $On(B, Table)$ $On(C, B)$ $On(A, Table) \wedge On(B, Table) \wedge On(C, B)$	$On(B, A)$ $On(A, C)$ $On(C, Table)$ $Clear(B)$ $Clear(Table)$

init



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

goal



$On(A, Table) \wedge On(B, Table) \wedge On(C, B)$

	PC	D	A
$move(x, y, z)$	$On(x, y)$ $Clear(x)$ $Clear(z)$	$On(x, y)$ $Clear(z)$	$On(x, z)$ $Clear(y)$ $Clear(Table)$

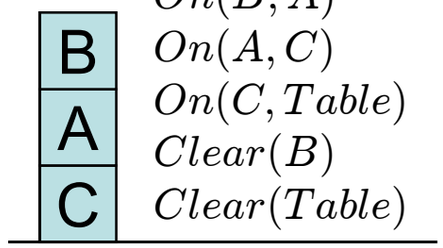
STRIPS(*init*,*goals*)

1. *state* = *init*, *plan* = [], *stack* = []
2. Push *goals* on *stack*
3. Repeat until *stack* = []
   - If *top* = *state* Pop *stack*
   - If *top* = conjunctive then Push subgoals onto *stack*.
   - If *top* = simple literal
     - Choose operator O where A_O = *top*
     - Replace *top* with operator O
     - Push preconditions of O onto *stack*
   - If top = operator O then Pop
     - *state* = apply(O,*state*)
     - *plan* = [*plan*, O]

STRIPS: Solve Blocks World

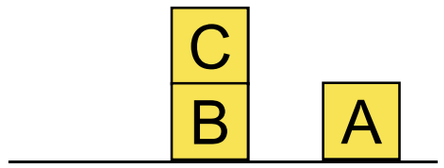
	Stack:	State:
7	$Clear(Table)$ $move(A, C, Table)$ $On(B, Table)$ $On(C, B)$ $On(A, Table) \wedge On(B, Table) \wedge On(C, B)$	$On(B, Table)$ $Clear(A)$ $On(A, C)$ $On(C, Table)$ $Clear(B)$ $Clear(Table)$
8	$move(A, C, Table)$ $On(B, Table)$ $On(C, B)$ $On(A, Table) \wedge On(B, Table) \wedge On(C, B)$	$On(B, Table)$ $Clear(A)$ $On(A, C)$ $On(C, Table)$ $Clear(B)$ $Clear(Table)$

init



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

goal



$On(A, Table) \wedge On(B, Table) \wedge On(C, B)$

	PC	D	A
$move(x, y, z)$	$On(x, y)$ $Clear(x)$ $Clear(z)$	$On(x, y)$ $Clear(z)$	$On(x, z)$ $Clear(y)$ $Clear(Table)$

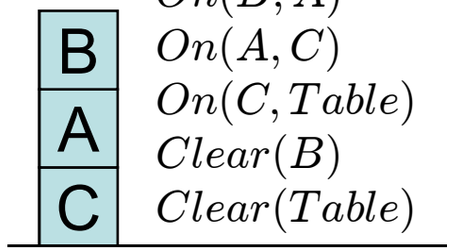
STRIPS(*init*,*goals*)

1. *state* = *init*, *plan* = [], *stack* = []
2. Push *goals* on *stack*
3. Repeat until *stack* = []
   - If *top* = *state* Pop *stack*
   - If *top* = conjunctive then Push subgoals onto *stack*.
   - If *top* = simple literal
     - Choose operator O where A_O = *top*
     - Replace *top* with operator O
     - Push preconditions of O onto *stack*
   - If top = operator O then Pop
     - *state* = apply(O,*state*)
     - *plan* = [*plan*, O]

STRIPS: Solve Blocks World

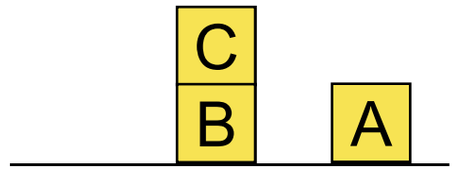
	Stack:	State:
7	$Clear(C)$ $move(A, C, Table)$ $On(B, Table)$ $On(C, B)$ $On(A, Table) \wedge On(B, Table) \wedge On(C, B)$	$On(B, Table)$ $Clear(A)$ $On(A, C)$ $On(C, Table)$ $Clear(B)$ $Clear(Table)$
8	$move(A, C, Table)$ $On(B, Table)$ $On(C, B)$ $On(A, Table) \wedge On(B, Table) \wedge On(C, B)$	$On(B, Table)$ $Clear(A)$ $On(A, C)$ $On(C, Table)$ $Clear(B)$ $Clear(Table)$
9	$On(B, Table)$ $On(C, B)$ $On(A, Table) \wedge On(B, Table) \wedge On(C, B)$	$On(A, Table)$ $Clear(C)$ $On(B, Table)$ $Clear(A)$ $On(C, Table)$ $Clear(B)$ $Clear(Table)$

init



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

goal



$On(A, Table) \wedge On(B, Table) \wedge On(C, B)$

	PC	D	A
$move(x, y, z)$	$On(x, y)$ $Clear(x)$ $Clear(z)$	$On(x, y)$ $Clear(z)$	$On(x, z)$ $Clear(y)$ $Clear(Table)$

STRIPS(*init*,*goals*)

1. *state* = *init*, *plan* = [], *stack* = []
2. Push *goals* on *stack*
3. Repeat until *stack* = []
   - If *top* = *state* Pop *stack*
   - If *top* = conjunctive then Push subgoals onto *stack*.
   - If *top* = simple literal
     - Choose operator O where A_O = *top*
     - Replace *top* with operator O
     - Push preconditions of O onto *stack*
   - If top = operator O then Pop
     - *state* = apply(O,*state*)
     - *plan* = [*plan*, O]

STRIPS: Solve Blocks World

Stack: | State:

9

$On(B, Table)$
$On(C, B)$
$On(A, Table) \wedge On(B, Table) \wedge On(C, B)$

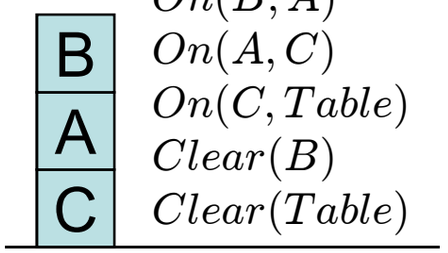
$On(A, Table)$
$Clear(C)$
$On(B, Table)$
$Clear(A)$
$On(C, Table)$
$Clear(B)$
$Clear(Table)$

10

$On(C, B)$
$On(A, Table) \wedge On(B, Table) \wedge On(C, B)$

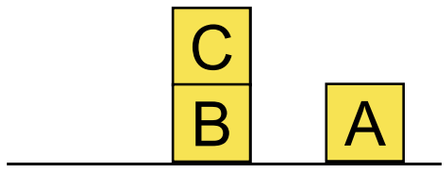
$On(A, Table)$
$Clear(C)$
$On(B, Table)$
$Clear(A)$
$On(C, Table)$
$Clear(B)$
$Clear(Table)$

init



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

goal



$On(A, Table) \wedge On(B, Table) \wedge On(C, B)$

	PC		D		A
$move(x, y, z)$	$On(x, y)$ $Clear(x)$ $Clear(z)$		$On(x, y)$ $Clear(z)$		$On(x, z)$ $Clear(y)$ $Clear(Table)$

STRIPS(*init*,*goals*)
1. *state* = *init*, *plan* = [], *stack* = []
2. Push *goals* on *stack*
3. Repeat until *stack* = []
   - If *top* = *state* Pop *stack*
   - If *top* = conjunctive then Push subgoals onto *stack*.
   - If *top* = simple literal
     - Choose operator O where A_O = *top*
     - Replace *top* with operator O
     - Push preconditions of O onto *stack*
   - If top = operator O then Pop
     - *state* = apply(O,*state*)
     - *plan* = [*plan*, O]

STRIPS: Solve Blocks World

Stack: | State:

10

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

$On(C, B)$

$On(A, Table) \wedge On(B, Table) \wedge On(C, B)$

11

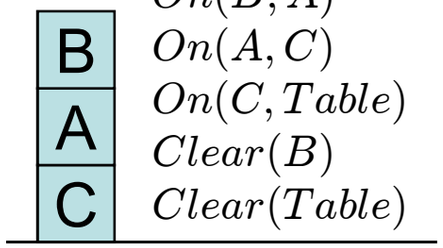
$On(A, Table)$
$Clear(C)$
$On(B, Table)$
$Clear(A)$
$On(C, Table)$
$Clear(B)$
$Clear(Table)$

$On(C, Table)$
$Clear(C)$
$Clear(B)$
$move(C, Table, B)$

$On(A, Table) \wedge On(B, Table) \wedge On(C, B)$

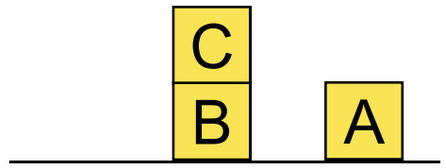
---

init



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

goal



$On(A, Table) \wedge On(B, Table) \wedge On(C, B)$

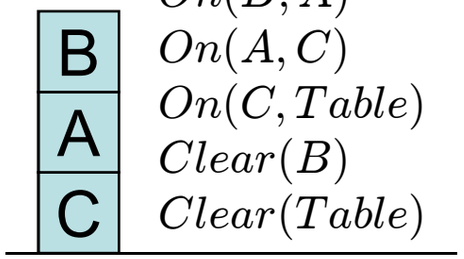
	PC	D	A
$move(x, y, z)$	$On(x, y)$ $Clear(x)$ $Clear(z)$	$On(x, y)$ $Clear(z)$	$On(x, z)$ $Clear(y)$ $Clear(Table)$

STRIPS(*init*,*goals*)
1. *state* = *init*, *plan* = [], *stack* = []
2. Push *goals* on *stack*
3. Repeat until *stack* = []
   - If *top* = *state* Pop *stack*
   - If *top* = conjunctive then Push subgoals onto *stack*.
   - If *top* = simple literal
     - Choose operator O where A_O = *top*
     - Replace *top* with operator O
     - Push preconditions of O onto *stack*
   - If top = operator O then Pop
     - *state* = apply(O,*state*)
     - *plan* = [*plan*, O]

STRIPS: Solve Blocks World

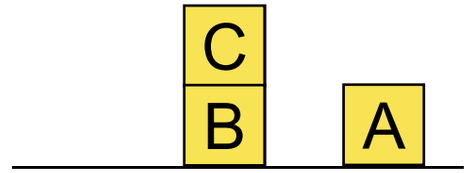
Stack:	State:
10	$On(A,Table)$ $Clear(C)$ $On(B,Table)$ $Clear(A)$ $On(C,Table)$ $Clear(B)$ $Clear(Table)$
$On(C,B)$ $On(A,Table) \wedge On(B,Table) \wedge On(C,B)$	
11	$On(A,Table)$ $Clear(C)$ $On(B,Table)$ $Clear(A)$ $On(C,Table)$ $Clear(B)$ $Clear(Table)$
$On(C,Table)$ $Clear(C)$ $Clear(B)$ $move(C,Table,B)$ $On(A,Table) \wedge On(B,Table) \wedge On(C,B)$	
12	$On(A,Table)$ $Clear(C)$ $On(B,Table)$ $Clear(A)$ $On(C,Table)$ $Clear(B)$ $Clear(Table)$
$move(C,Table,B)$ $On(A,Table) \wedge On(B,Table) \wedge On(C,B)$	

init



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

goal



$On(A,Table) \wedge On(B,Table) \wedge On(C,B)$

	PC	D	A
$move(x,y,z)$	$On(x,y)$ $Clear(x)$ $Clear(z)$	$On(x,y)$ $Clear(z)$	$On(x,z)$ $Clear(y)$ $Clear(Table)$

STRIPS(*init*,*goals*)

1. *state* = *init*, *plan* = [], *stack* = []
2. Push *goals* on *stack*
3. Repeat until *stack* = []
   - If *top* = *state* Pop *stack*
   - If *top* = conjunctive then Push subgoals onto *stack*.
   - If *top* = simple literal
     - Choose operator O where A_O = *top*
     - Replace *top* with operator O
     - Push preconditions of O onto *stack*
   - If top = operator O then Pop
     - *state* = apply(O,*state*)
     - *plan* = [*plan*, O]

STRIPS: Solve Blocks World

Stack: | State:

12

$move(C, Table, B)$
$On(A, Table) \wedge On(B, Table) \wedge On(C, B)$

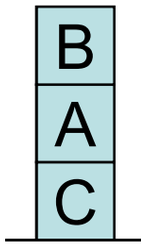
$On(A, Table)$
$Clear(C)$
$On(B, Table)$
$Clear(A)$
$On(C, Table)$
$Clear(B)$
$Clear(Table)$

13

$On(A, Table) \wedge On(B, Table) \wedge On(C, B)$

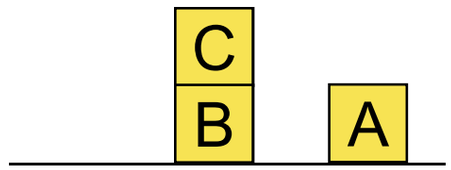
$On(C, B)$
$On(A, Table)$
$Clear(C)$
$On(B, Table)$
$Clear(A)$
$Clear(Table)$

init



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

goal



$On(A, Table) \wedge On(B, Table) \wedge On(C, B)$

	PC	D	A
$move(x, y, z)$	$On(x, y)$ $Clear(x)$ $Clear(z)$	$On(x, y)$ $Clear(z)$	$On(x, z)$ $Clear(y)$ $Clear(Table)$

STRIPS(*init*,*goals*)
1. *state* = *init*, *plan* = [], *stack* = []
2. Push *goals* on *stack*
3. Repeat until *stack* = []
   - If *top* = *state* Pop *stack*
   - If *top* = conjunctive then Push subgoals onto *stack*.
   - If *top* = simple literal
     - Choose operator O where A_O = *top*
     - Replace *top* with operator O
     - Push preconditions of O onto *stack*
   - If top = operator O then Pop
     - *state* = apply(O,*state*)
     - *plan* = [*plan*, O]

STRIPS: Solve Blocks World

Stack: State:

12

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

$move(C, Table, B)$
$On(A, Table) \wedge On(B, Table) \wedge On(C, B)$

13

$On(A, Table) \wedge On(B, Table) \wedge On(C, B)$

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

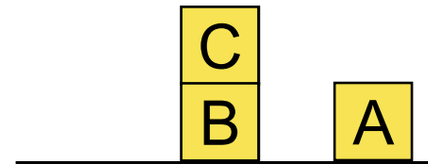
Remember the Plan?

$move(B, A, Table)$
$move(A, C, Table)$
$move(C, Table, B)$

init

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

goal



$On(A, Table) \wedge On(B, Table) \wedge On(C, B)$

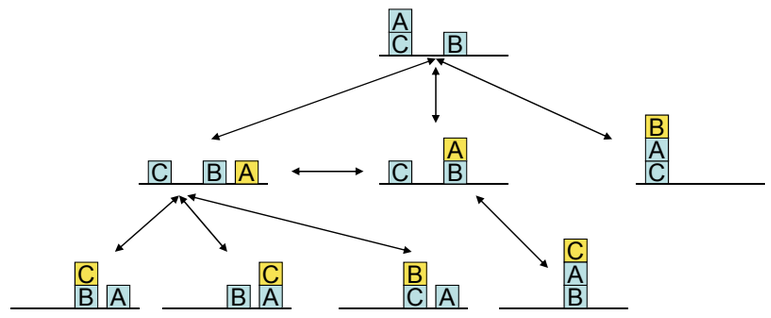
	PC		D		A
$move(x, y, z)$	$On(x, y)$ $Clear(x)$ $Clear(z)$		$On(x, y)$ $Clear(z)$		$On(x, z)$ $Clear(y)$ $Clear(Table)$

STRIPS(*init*,*goals*)
1. *state* = *init*, *plan* = [], *stack* = []
2. Push *goals* on *stack*
3. Repeat until *stack* = []
   - If *top* = *state* Pop *stack*
   - If *top* = conjunctive then Push subgoals onto *stack*.
   - If *top* = simple literal
     - Choose operator O where A_O = *top*
     - Replace *top* with operator O
     - Push preconditions of O onto *stack*
   - If top = operator O then Pop
     - *state* = apply(O,*state*)
     - *plan* = [*plan*, O]

STRIPS: Summing it up

- Uses literals to represent the world
- State-based view of time
- "Most predicates are unaffected by a single action"

- Can be easily used for planning and connect to search algorithms.
- State space searches



Did STRIPS Solve Our Problems?

Frame Problem?

Yes!

Ramification Problem?

No

Separate ground facts from inferred facts

Only ground facts are carried over from state to state

Potentially inefficient

Planning method?

yes, linear serach

Is it good?