


Part II
Methods of AI

Chapter 4
Planning

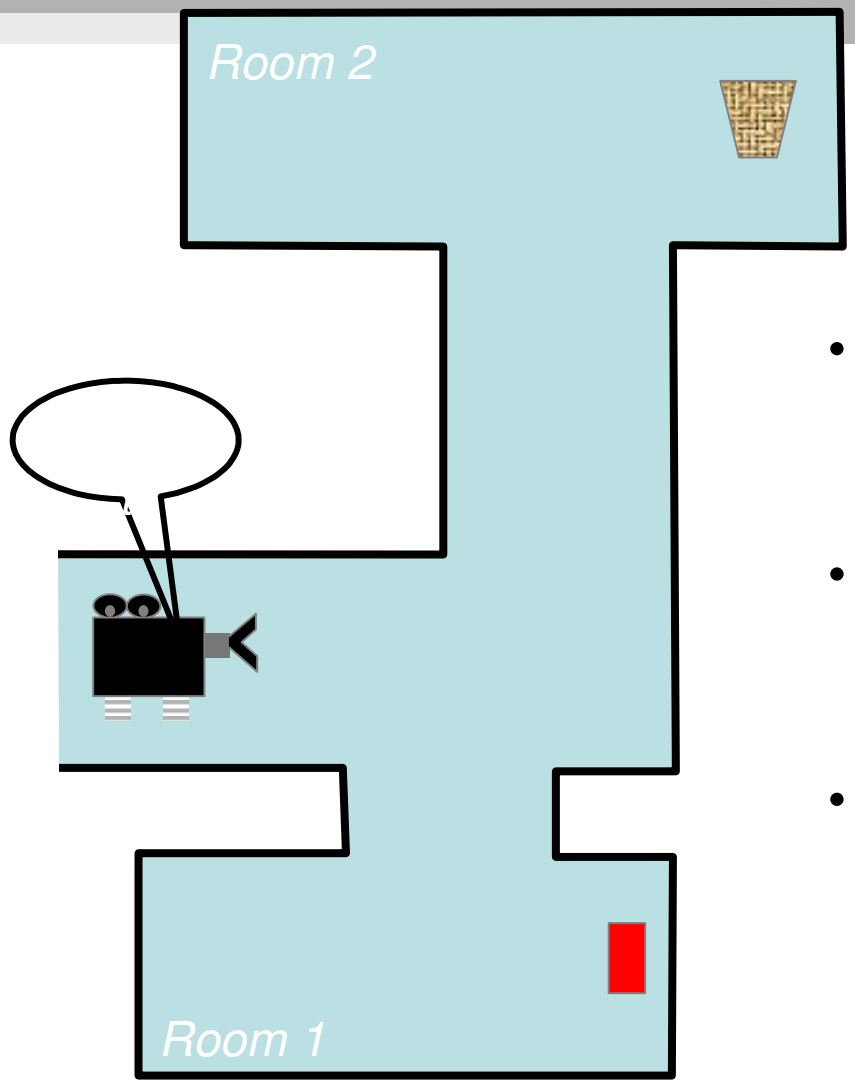
UNIVERSITÄT DES SAARLANDES

DFKI

What is AI Planning ?

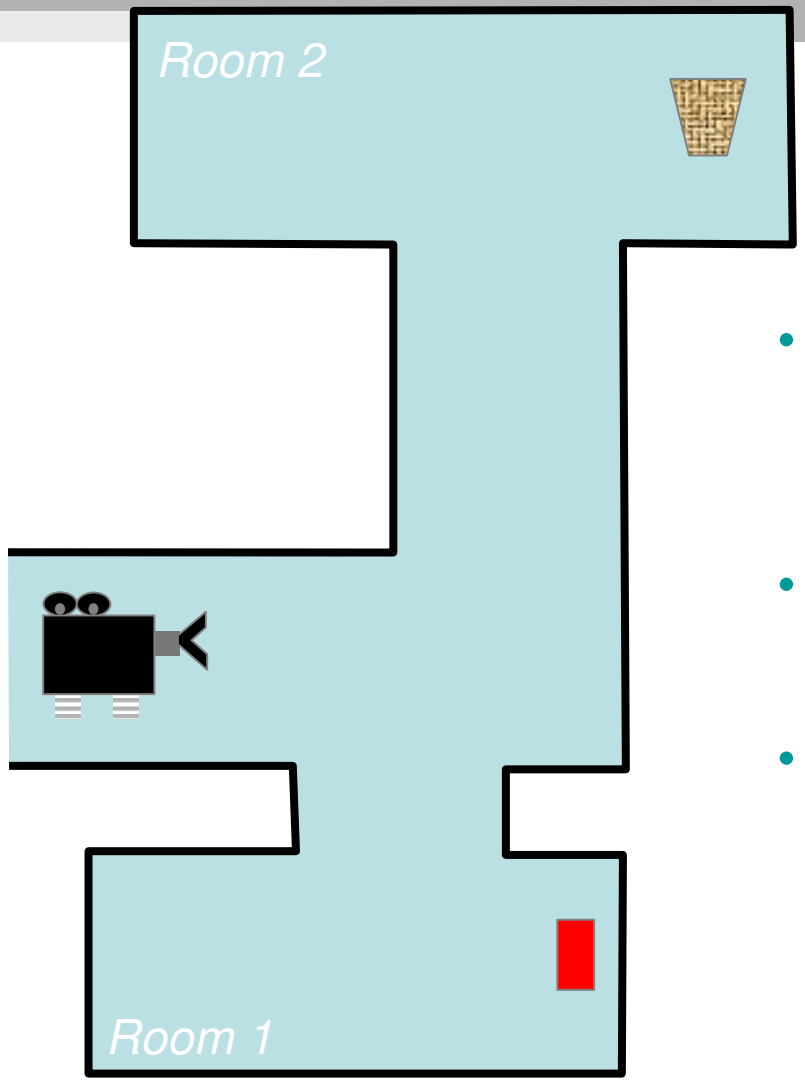
- Generate sequences of actions to perform tasks and achieve objectives
- Until recently, AI planning was essentially a theoretical endeavor: It‘s now becoming useful in many industrial applications
- **Example application areas**
  -
  -
  -
  -

Planning Involves



- Given knowledge about task domain (*actions*)
- Given *problem* specified by initial state configuration and goals to achieve
- Agent tries to find a solution, i.e. a sequence of actions that solves a problem

Notions



Go to the basket

- **Plan**
- **Operators**
- **Planner**

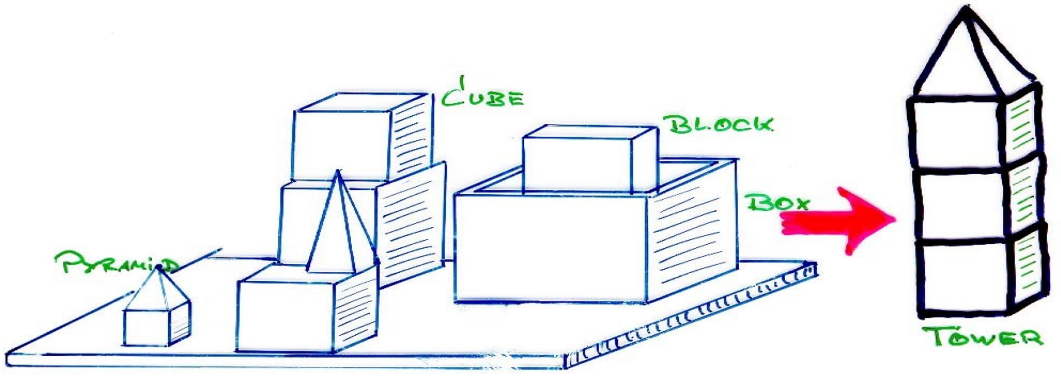
MOTIVATION

PLAN

REPRÄSENTATION

Planning in the Blocks World

Planen in der Klötzchen-Welt



The Blocks World in Reality!

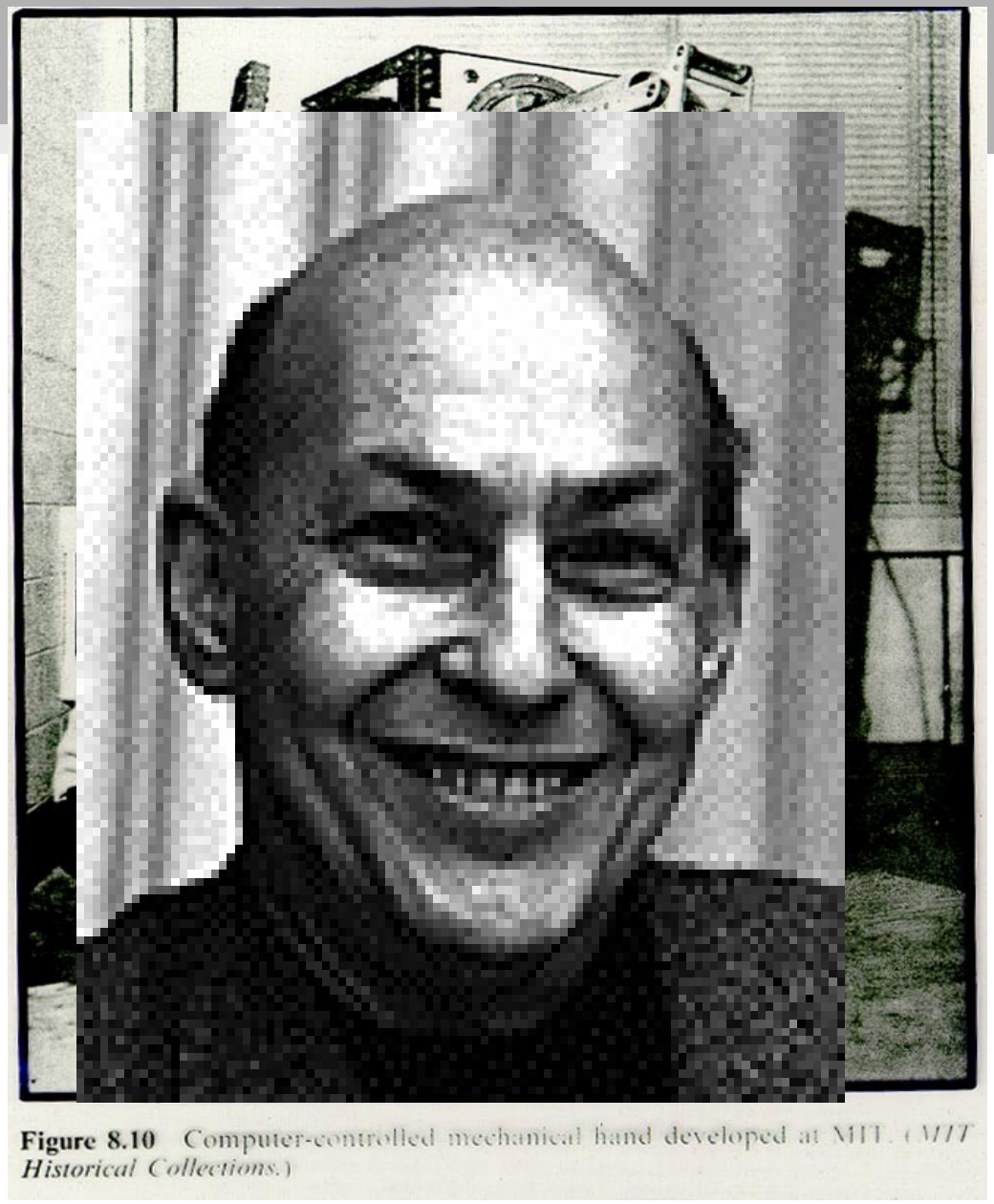


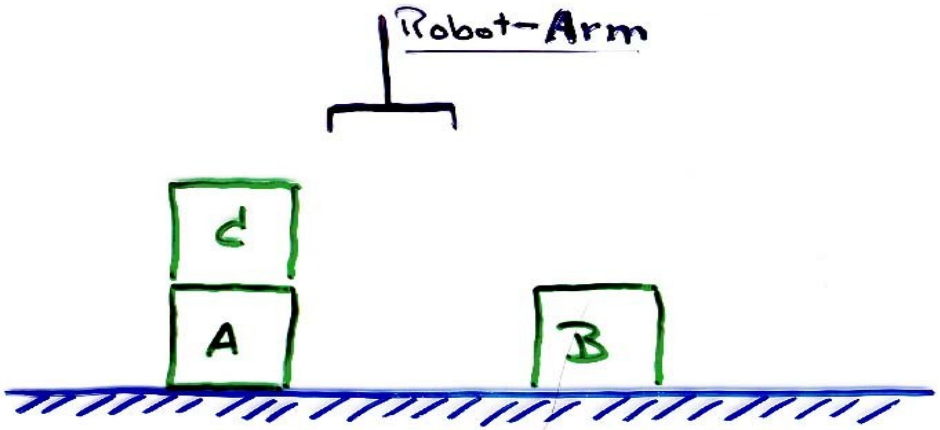
Figure 8.10 Computer-controlled mechanical hand developed at MIT. (*MIT Historical Collections.*)

. . . . . and now!

UNIVERSITÄT DES SAARLANDES

DFKI

Ausgangs-Zustand:

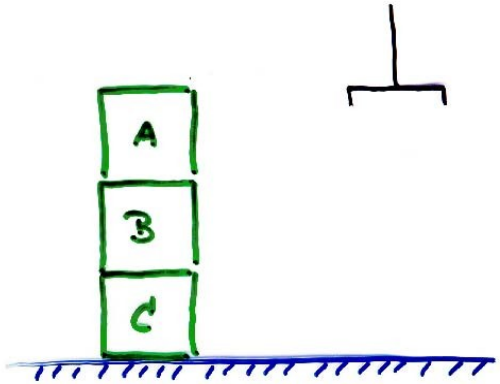


Zustands-Beschreibung:

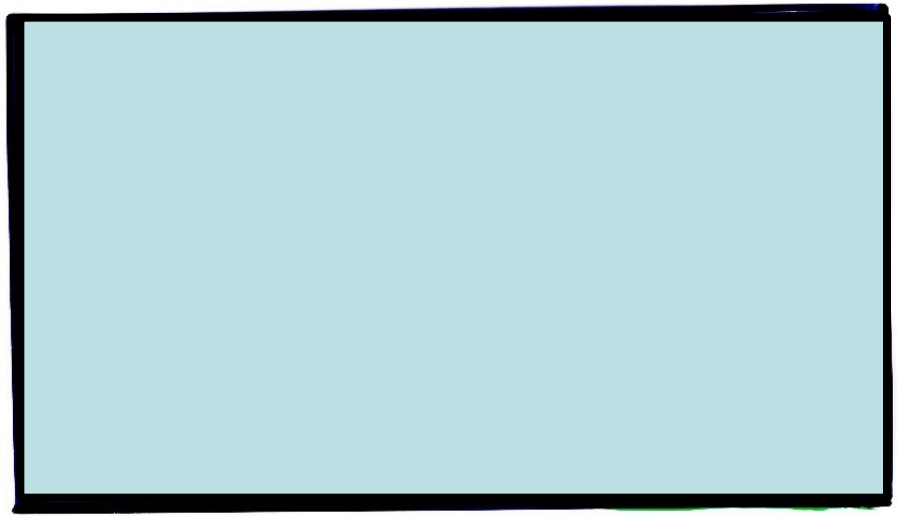
(state description)

- Clear (B)
- Clear (C)
- On (C, A)
- Ontable (A)
- Ontable (B)
- Hand Empty


ZIEL - ZUSTAND:



ZIEL: "ON(A,B) AND ON(B,C)"
(goal description)



AKTIONEN (actions):

(i) **PICKUP(x)**

PRECONDITION:	ONTABLE(x), CLEAR(x), HANDEMPTY
DELETE:	ONTABLE(x), CLEAR(x), HANDEMPTY
ADD:	HOLDING(x)

(ii) **PUTDOWN(x)**

PRECONDITION:	HOLDING(x)
DELETE:	HOLDING(x)
ADD:	ONTABLE(x), CLEAR(x), HANDEMPTY

(iii) **STACK(x, y)**

PRECONDITION:	HOLDING(x), CLEAR(y)
DELETE:	HOLDING(x), CLEAR(y)
ADD:	HANDEMPTY, ON(x, y), CLEAR(x)

(iv) **UNSTACK(x, y)**

PRECONDITION:	HANDEMPTY, CLEAR(x), ON(x, y)
DELETE:	HANDEMPTY, CLEAR(x), ON(x, y)
ADD:	HOLDING(x), CLEAR(y)

ZIEL - ZUSTAND:

The final Plan

PLAN:

(i) UNSTACK(D, A)
(ii) PUTDOWN(D)
(iii) PICKUP(B)
(iv) STACK(B, C)
(v) PICKUP(A)
(vi) STACK(A, B)


Repräsentation Eines Planes

- Spotty - Beispiel
- Unterbrechung / Fehler

Dreiecks-Tabelle

	0	1	2	3	4	5	6
1	*HANDEMPTY* *CLEAR(C)* *ON(C,A)*	**unstack**(*C,A*)					
2		*HOLDING(C)*	**putdown**(*C*)				
3	*ONTABLE(B)* *CLEAR(B)*		*HANDEMPTY*	**pickup**(*B*)			
4			*CLEAR(C)*	*HOLDING(B)*	**stack**(*B,C*)		
5	*ONTABLE(A)*	*CLEAR(A)*			*HANDEMPTY*	**pickup**(*A*)	
6					*CLEAR(B)*	*HOLDING(A)*	**stack**(*A,B*)
7					*ON(B,C)*		*ON(A,B)*

Fig. 7.4 A triangle table.

Plansequenz

Precondition

Kern Einer Dreieckstabelle

- 4-Kern

SHAKEY:

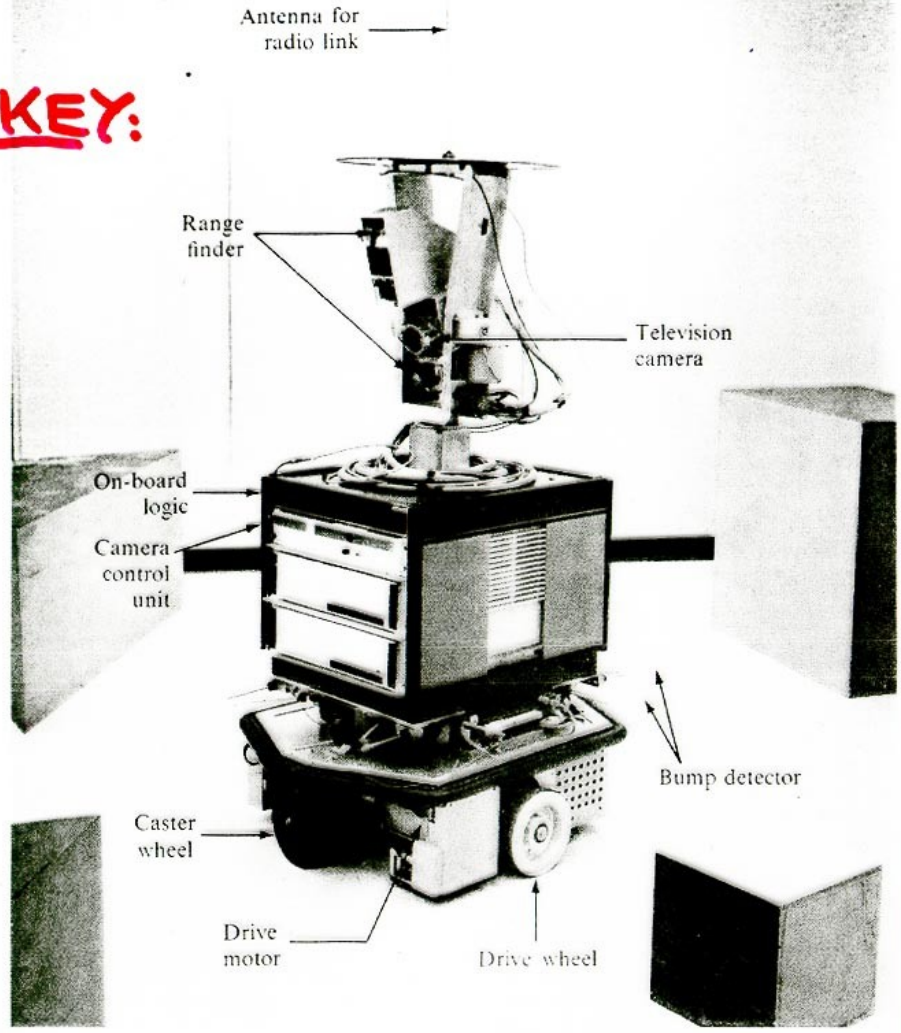
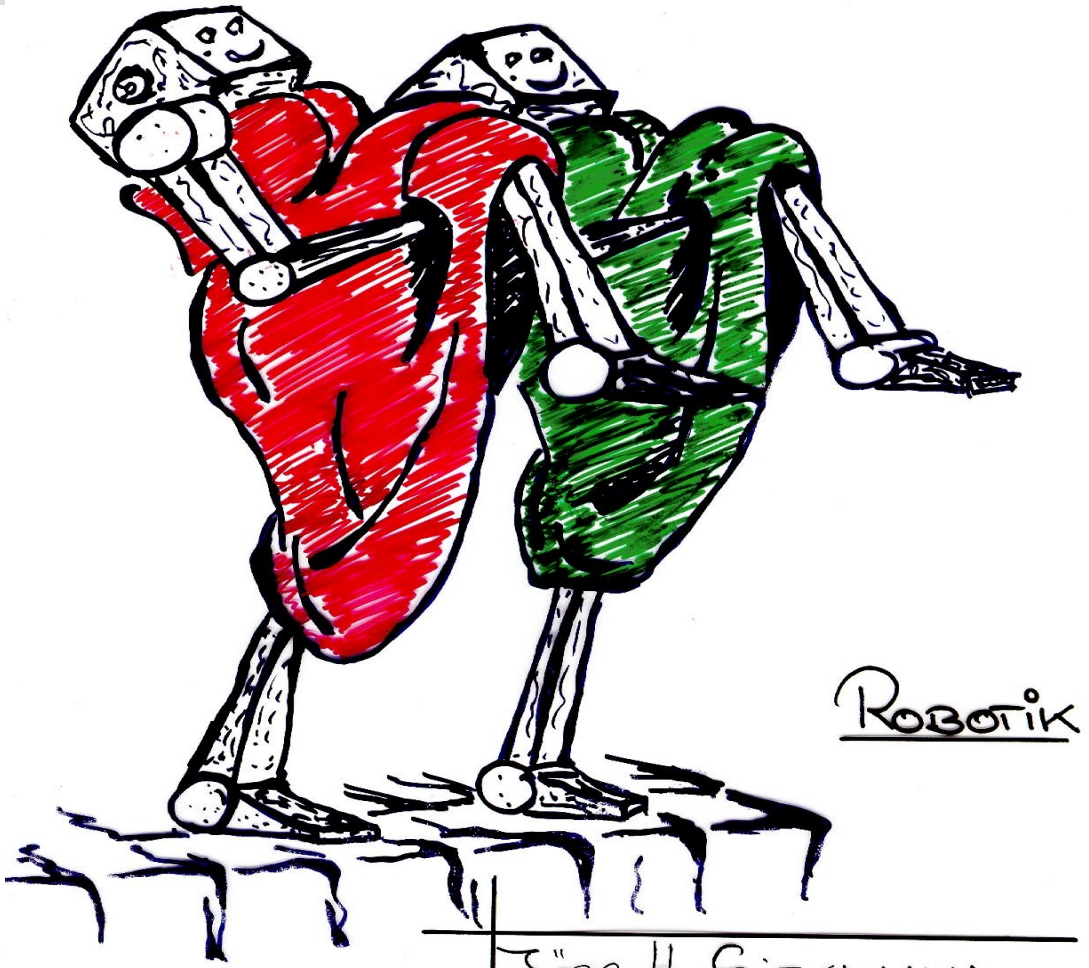


Figure 8.3 Shakey, the robot developed at SRI. (*Stanford Research Institute.*)

UNIVERSITÄT DES SAARLANDES

DFKI

MENSCH MASCHINE MENSCH



JÖRG H. SIEKMANN
DFKI + Univ. Saarbrücken

Part II: Methods of AI

Chapter 4 - Planning

4.1 State Space Planning

4.2 Partial Order Planning

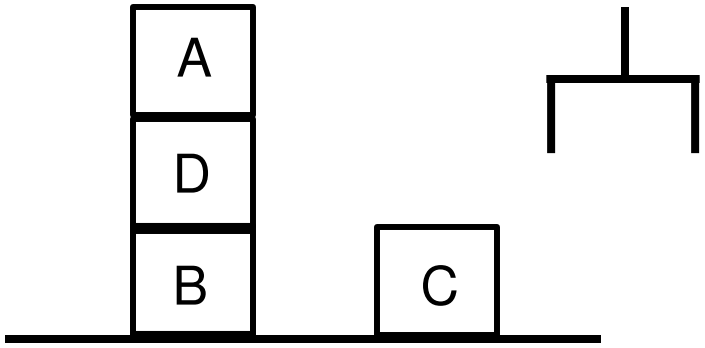
4.3 Planning in the Real World

What is a Planning Problem?

A planning problem is given by:

an **initial state** and a **goal state.**

GOAL:




```
Ontable (B)
Ontable (C)
On (D,B)
On (A,D)
Clear (A)
Clear (C)
Handempty
```

For a transition there are certain **operators** available.

`PICKUP(x)`	picking up x from the table
`PUTDOWN(x)`	putting down x on the table
`STACK(x,y)`	putting x on y
`UNSTACK(x,y)`	picking up x from y

→Operators in blocks world!

→NOW:

- Formalise Operators!
- Find a plan!

UNIVERSITÄT DES SAARLANDES

DFKI

Simplifications

- **Atomic time**: execution time of action not interruptable. Simultaneously executed actions not possible (for now)
- **Deterministic effects**: effect is a deterministic function of actions and state into states when the action is executed.
- **Omniscience**: agent has complete knowledge of initial state and of its own actions.
- **Sole cause of change**: only the agent´s actions change the world.
- **STRIPS assumption**: literals not mentioned remain unchanged:

  The “Frame Problem”

- **Planner** has complete information

The classical Planning Framework

Example: Travel Scenario

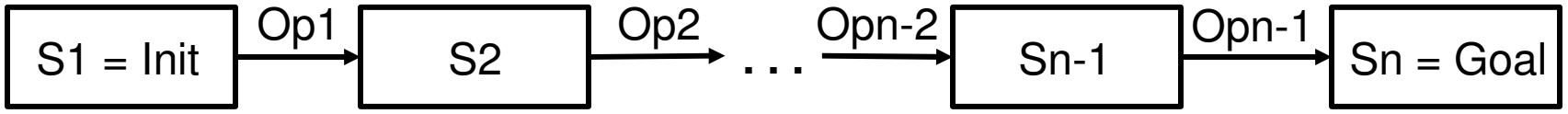
Two phases: Plan Generation and Plan Execution

Given: Initial state, Goal State and a Set of Operators

Task: Find planning operators which transform initial state into goal state

Describe **states** in a formal language like Predicate Logic

A Plan is a sequence of operators that transforms the initial state into the goal state



The STRIPS Language

- **A subset of first-order logic:**
  - predicate symbols (chosen for the particular domain)
  - constant symbols (chosen for the particular domain)
  - variable symbols
  - no function symbols
- **Atom: expression $p(t_1, ..., t_n)$**
  - p is a predicate symbol
  - each t_i is a term

The STRIPS Language

- **A Literal:**

  Is an atom $p(t_1, ..., t_n)$, called a *positive literal* or a negated atom $\sim p(t_1, ..., t_n)$, called a *negative literal*

- A **conjunct:**

  is represented either by a comma or a $\wedge$:

  $p_1(t_1, ..., t_n), \sim p_2(t_1, ..., t_n), p_3(t_1, ..., t_n)$

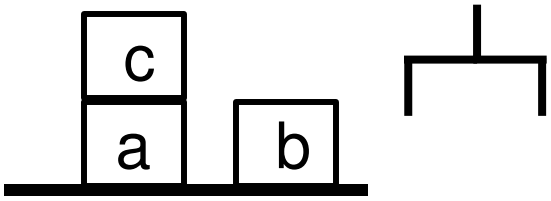
- For now, we won't have any disjunctions, implications, or quantifiers

Representing States of the World

- **State:**

  a consistent assignment of TRUE or FALSE to every literal in the universe

- **State Description:**
  - a set of ground literals that are all taken to be TRUE



`on(c,a),ontable(a),clear(c),`
`ontable(b),clear(b),handempty`

  - ➢ The negation of these literals are taken to be false
  - ➢ Truth values of other ground literals are unknown

  - Note: in some books and papers, a state is restricted to contain only positive literals (more on this later)

STRIPS Operators

- **A STRIPS Operator**:

```
name(v1, v2, ..., vn)
   Preconditions:  literal1, literal2, ..., literaln
   Effects:        literal1, literal2, ..., literalm
```

Example:

```
unstack(?x,?y)
   Preconditions:    on(?x,?y), clear(?x), handempty
   Effects:           ~on(?x,?y), ~clear(?x), ~handempty
                     holding(?x), clear(?x)
```

- **Operator Instance:** replacement of variables by constants

STRIPS Operators

- **Ground instance:** replace all variables by constants

```
unstack(c,a)
  Preconditions:    on(c,a), clear(c), handempty
  Effects:          ~on(c,a), ~clear(c), ~handempty,
                    holding(c), clear(a)
```

- **Delete:**

For a ground instance operator *O:*

Del (*O*) = {*p* |: ~*p* is in effects(*O*)}

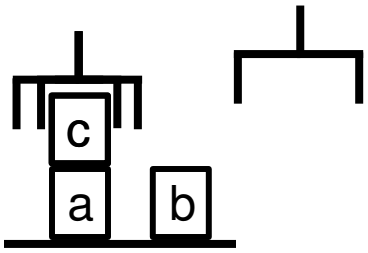
```
Del(unstack(c,a)) =
   {on(c,a),clear(c),handempty}
```

STRIPS Operators

- **Applicable:** If all preconditions of a ground instance are true (i.e., they occur) in a state description *S*, then *O* is applicable to *S*
- **Application:**

  Applying operator *O* to *S* produces the *successor state description:*

$$\text{Result}(S,O) = (S - \text{Del}(O)) \cup \text{Effects}(O)$$




```
on(c,a), ontable(a), clear(c),
   ontable(b), clear(b),handempty
```

```
unstack(c,a)
  Preconditions: on(c,a), clear(c), handempty
  Effects:       ~on(c,a), ~clear(c), ~handempty,
                 holding(c), clear(a)
```

```
ontable(a), ontable(b), clear(b), ~on(c,a),
   ~clear(c), ~handempty, holding(c), clear(a)
```

UNIVERSITÄT DES SAARLANDES

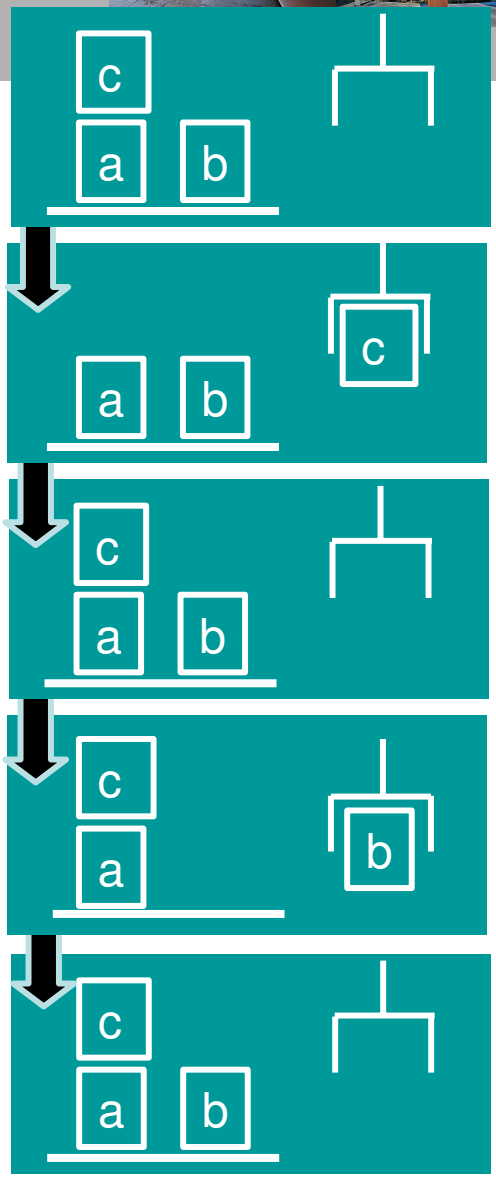
Example: The Blocks World

unstack(?x, ?y)
```
Pre: on(?x,?y), clear(?x), handempty
Eff: ~on(?x,?y), ~clear(?x), ~handempty,
     holding(?x), clear(?y)
```

stack(?x, ?y)
```
Pre: holding(?x), clear(?y)
Eff: ~holding(?x), ~clear(?y),
     on(?x,?y), clear(?x), handempty
```

pickup(?x)
```
Pre: ontable(?x), clear(?x), handempty
Eff: ~ontable(?x), ~clear(?x),
     ~handempty, holding(?x)
```

putdown(?x)
```
Pre: holding(?x)
Eff: ~holding(?x), ontable(?x),
     clear(?x), handempty
```

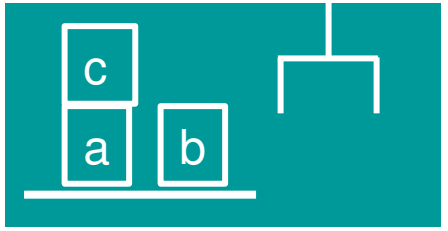


STRIPS Operators:

alternative Formulation without Negation (~)

- States contain only atoms (positive literals)

```
on(c,a), ontable(a)
clear(c), ontable(b)
clear(b), handempty(
        )
```



- STRIPS operators use a delete list instead of negated effects

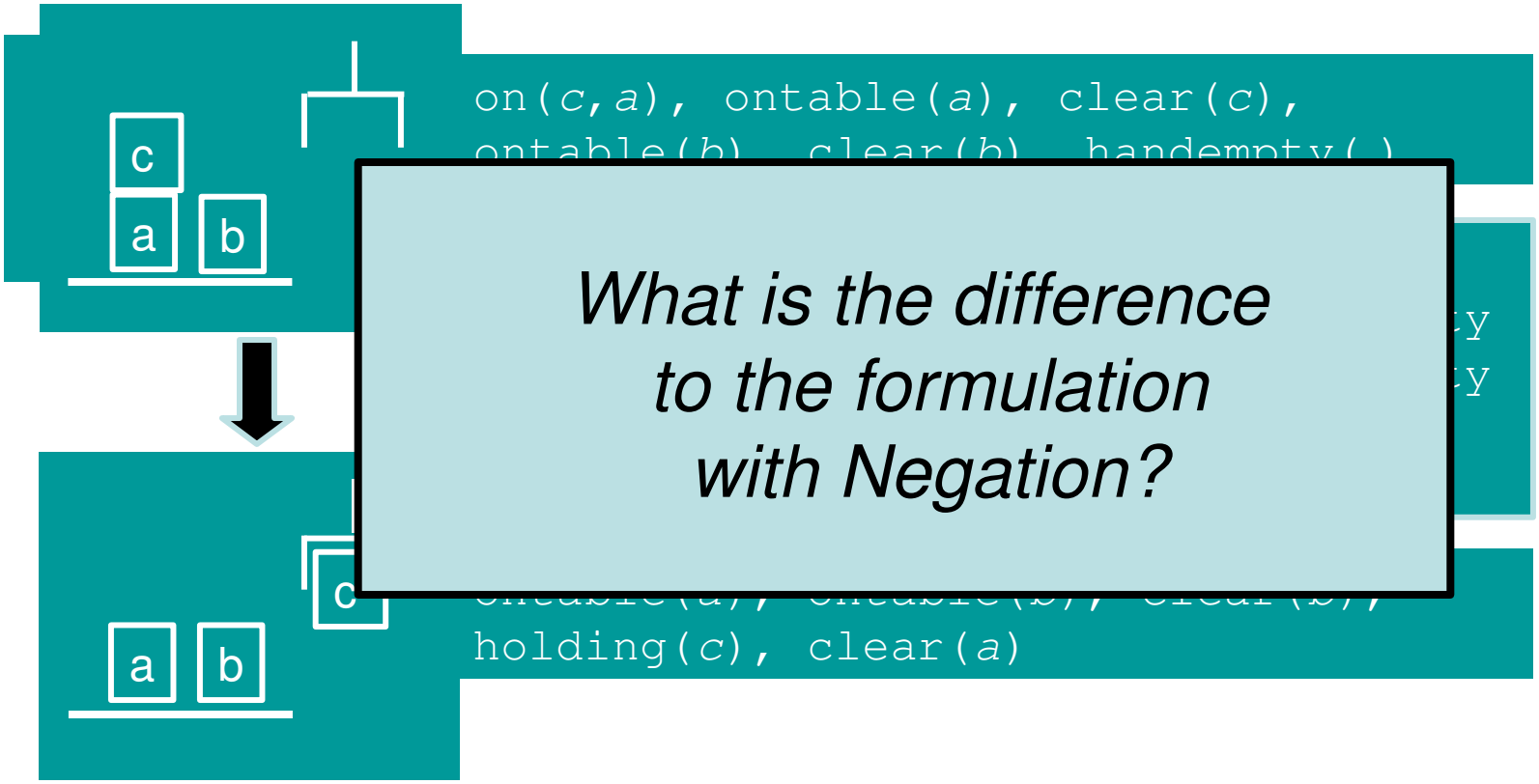
```
name(v1, ..., vn)
    Pre:  atom, ..., atom
    Add:  atom, ..., atom
    Del:  atom, ..., atom
```

```
unstack(?x,?y)
 Pre: on(?x,?y), clear(?x), handempty
 Del: on(?x,?y), clear(?x), handempty,
 Add: holding(?x), clear(?y)
```

STRIPS Operators
alternative Formulation

- If *O* is applicable to *S*, then

  result(*S*,*O*) = (*S* - Del(*O*)) ∪ Add(*O*)



What is the difference to the formulation with Negation?

Plans

- **STRIPS planning domain**:
  - a language *L* (choose the constant symbols)
  - a set of planning operators (e.g., the blocks-world operators)
- **Plan:** a sequence $P = (o_1, o_2, ..., o_k)$ of ground instances of operators

```
unstack(c,a), putdown(c), pickup(a), stack(a,c)
```

- Each o_i is called a step of *P*

Plans

- **Result(*S*,*P*)**

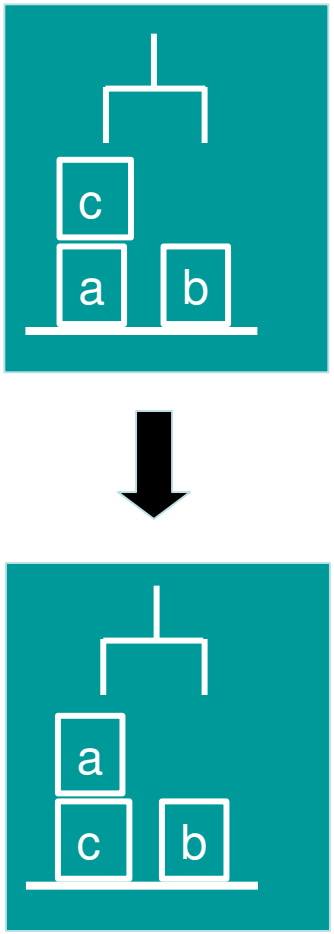
  is the state produced by starting with the state S, and applying o_1,

  then o_2,… ..., and then finally o_k

Formal definition:

Case 1: $k = 1$ (i.e., $P = (o1)$)
If result(S, $o1$) is defined, then
result(S,P) = result(S,$o1$)

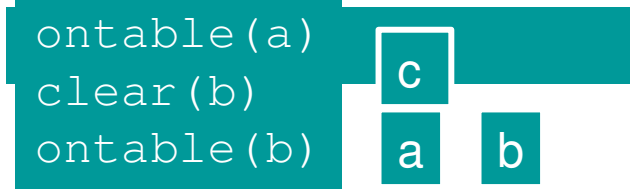
Case 2: $k > 1$ Let $P' = (o1, \ldots, ok\text{-}1)$
If result(S, P') is defined, then
result(S, P) = result(result(S, P'), P)



Planning Problems

- **STRIPS planning problem**
  - a triple $R = (i, g, O)$ where
    - ➢
    - ➢
    - ➢

```
clear(c)
on(c,a)
ontable(a)
clear(b)
ontable(b)
handempty
```



- P is a correct plan for R if g is true in result (i, P)

```
on(c,a)
ontable(a)
```


```
unstack(c,a), putdown(c), pickup(a), stack(a,c)
```

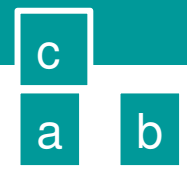
```
unstack(c,a), putdown(c), pickup(a), stack(a,b)
```

```
pickup(a), stack(a,c), unstack(c,a), putdown(c)
```

Properties of Plans

- **Irredundancy**
  - No proper subsequence of *P* is a correct plan for *R*
- **Optimality**
  - If *P* is a plan returned by the algorithm, then no correct plan for *R* has lower cost than *P*

```
clear(c)
on(c,a)
ontable(a)
clear(b)
ontable(b)
handempty
```




```
on(a,b)
ontable(c)
```


```
unstack(c,a), putdown(c), pickup(a), stack(a,c)
```

```
unstack(c,a), putdown(c), pickup(a),
putdown(a), pickup(a), stack(a,c)
```

UNIVERSITÄT DES SAARLANDES

Questions

- Suppose we know the following:
  - CLEAR(A, s), CLEAR(B, s), CLEAR(C, s)
- Then which of the following statements can we infer?
  - CLEAR(A, do(trans(A, D, B), s))
  - ON(A, B, do(trans(A, D, B), s))
  - CLEAR(C, do(trans(A, D, B), s))

$$[CLEAR(x, s) \wedge CLEAR(z, s) \wedge ON(x, y, z) \wedge DIFF(x, z)]$$
$$\Rightarrow [CLEAR(x, do[trans(x, y, z), s])$$
$$\wedge CLEAR(y, do[trans(x, y, z), s])$$
$$\wedge ON(x, z, do[trans(x, y, z), s])].$$

(variables in assertions have implicit universal quantifications)

UNIVERSITÄT DES SAARLANDES

The "**Frame Problem**"

- Need to describe both what changes and what *doesn't* change:

  either implicitly as in STRIPS or

  explicitely with *frame axioms*

- One of the earliest solutions to the frame problem was the STRIPS planning algorithm
  - Specialized planning algorithm rather than general-purpose theorem prover
  - Leaves facts unchanged from one state to the next unless a planning operator explicitly changes them

The STRIPS Algorithm

while GOALS contains one or more literals not in S do

;; nondeterministic choice

g := any literal in GOALS that is not in S

if there is no operator whose effects match g then return FAIL

;; nondeterministic choice

o := any operator whose effects contain a literal a that unifies with g

δ := an mgu for a and g

if STRIPS(preconditions(δ o)) returns FAIL then return FAIL

o' := the instance of δ o whose preconditions are true in S

S := the result of applying o' to S

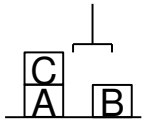
end while

end STRIPS

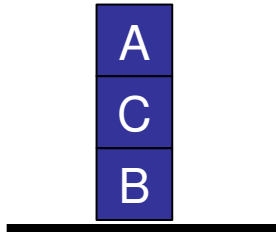
Example (1)

STATE DESCRIPTION

CLEAR(B)
CLEAR(C)
ON(C,A)
ONTABLE(A)
ONTABLE(B)
HANDEMPTY



GOAL STACK
ON(C,B) ∧ ***ON(A,C)***



GOAL STACK
ON(A,C)
ON(C,B)
ON(C,B) ∧ *ON(A,C)*

GOAL STACK
CLEAR(C) ∧ ***HOLDING(A)***
stack(A,C)
ON(C,B)
ON(C,B) ∧ *ON(A,C)*

Not a promising solution path

GOAL STACK
ON(C,B)
ON(A,C)
ON(C,B) ∧ *ON(A,C)*

GOAL STACK
CLEAR(B) ∧ ***HOLDING(C)***
stack(C,B)
ON(C,B)
ON(C,B) ∧ *ON(A,B)*

Continued next page

(from Nilsson 1980)

UNIVERSITÄT DES SAARLANDES

DFKI

Example (2)

Continued from previous page

GOAL STACK

HOLDING(C)
CLEAR(B)
CLEAR(B) ∧ HOLDING(C)
stack(C,B)
ON(C,B)
ON(C,B) ∧ ON(A,B)

Not a promising solution path

STATE DESCRIPTION	**GOAL STACK**
CLEAR(B)	***HANDEMPTY ∧ CLEAR (C) ∧ ON(C,Y)***
CLEAR(C)	***unstack(C, Y)***
ON(C,A)	*CLEAR(B) ∧HOLDING(C)*
ONTABLE(A)	*stack (C,B)*
ONTABLE(B)	*ON(A,C)*
HANDEMPTY	*ON(C,B) ∧ ON(A,C) ON(A,C)*

Continued next page

(from Nilsson 1980)

UNIVERSITÄT DES SAARLANDES

DFKI

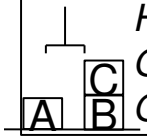
Example (3)

Continued from previous page

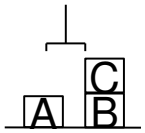
With {A/y} the top subgoal matches
The current state description. We can
Then apply unstack (C,A). Now the
Next two goals match also, so we can
Apply stack (C,B)

STATE DESCRIPTION	GOAL STACK
CLEAR(C)	*ON(A,C)*
CLEAR(A)	*ON(C,B) ∧ ON(A,C)*
ON(C,B)	
HANDEMPTY	
ONTABLE(A)	
ONTABLE(B)	

STATE DESCRIPTION	GOAL STACK
CLEAR(C)	***CLEAR(C) ∧ HOLDING (A)***
CLEAR(A)	***stack(A,C)***
ON(C,B)	*ON(C,B) ∧ ON(A,C)*
HANDEMPTY	
ONTABLE(A)	
ONTABLE(B)	



STATE DESCRIPTION	GOAL STACK
CLEAR(C)	***ONTABLE(A) ∧ CLEAR (A)***
CLEAR(A)	***∧ HANDEMPTY***
ON(C,B)	***pickup(A)***
HANDEMPTY	*CLEAR(C) ∧ HOLDING(A)*
ONTABLE(A)	*stack(A,C)*
ONTABLE(B)	*ON(C,B) ∧ ON(A,C)*



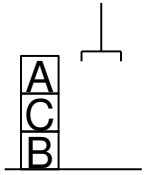
Now we can apply pickup(A),and
Then the next goal will be matched.
So we can apply stack(A,C). Now
The last remaining goal on the stack
In matched.

(from Nilsson 1980)

Continued next page

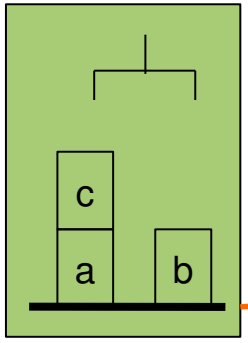
Example (4)

Continued from previous page

STATE DESCRIPTION	GOAL STACK
 *ON(A,C)* *ON(C,B)* *HANDEMPTY* *CLEAR(A)* *ONTABLE(B)*	***NIL***

(from Nilsson 1980)

Complete Search Space for Breadth-First search



CLEAR(A) ONTABLE(A)
CLEAR(B) ONTABLE(B)
CLEAR(C) ONTABLE(C)
HANDEMPTY

putdown(*B*) pickup(*B*) pickup(*C*) Putdown(*C*) pickup(*A*) putdown(*A*)

CLEAR(A) CLEAR(C) HOLDING(B) ONTABLE(A) ONTABLE(C)

CLEAR(A) CLEAR(B) HOLDING(C) ONTABLE(A) ONTABLE(B)

CLEAR(B) CLEAR(C) HOLDING(A) ONTABLE(B) ONTABLE(C)

stack(B, A) unstack(B, A) stack(C, A) unstack(C, A) stack(A, B) unstack(A, B)

stack(B, C) unstack(B, C) stack(C, B) unstack(C, B) stack(A, C) unstack(A, C)

CLEAR(A) ON(B, C) CLEAR(B) ONTABLE(A) ONTABLE(C) HANDEMPTY

CLEAR(C) ON(B, A) CLEAR(B) ONTABLE(A) ONTABLE(C) HANDEMPTY

CLEAR(A) ON(C, B) CLEAR(C) ONTABLE(A) ONTABLE(B) HANDEMPTY

CLEAR(B) ON(C, A) CLEAR(C) ONTABLE(A) ONTABLE(B) HANDEMPTY

CLEAR(B) ON(A, C) CLEAR(A) ONTABLE(B) ONTABLE(C) HANDEMPTY

CLEAR(C) ON(A, B) CLEAR(A) ONTABLE(B) ONTABLE(C) HANDEMPTY

pickup(*A*) putdown(*A*) pickup(*c*) putdown(*C*) pickup(*A*) putdown(*B*) pickup(*B*) putdown(*B*) pickup(*B*) putdown(*B*) pickup(*C*) putdown(*C*)

ON(B, C) CLEAR(B) HOLDING(A) ONTABLE(C)

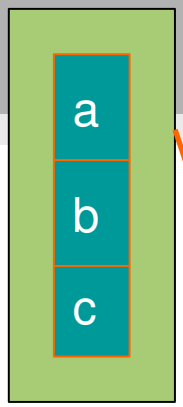
ON(B, C) CLEAR(B) HOLDING(A) ONTABLE(C)

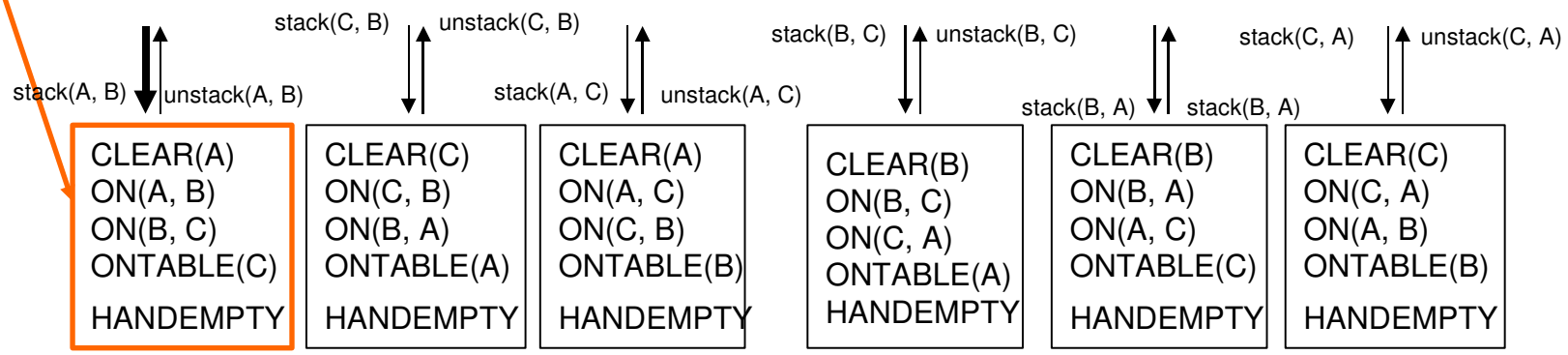
ON(B, C) CLEAR(B) HOLDING(A) ONTABLE(C)

ON(B, C) CLEAR(B) HOLDING(A) ONTABLE(C)

ON(B, C) CLEAR(B) HOLDING(A) ONTABLE(C)

ON(B, C) CLEAR(B) HOLDING(A) ONTABLE(C)

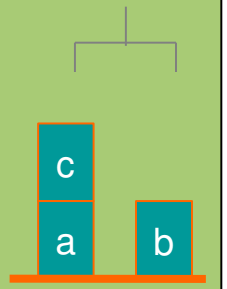

a
b
c


stack(A, B)
unstack(A, B)
stack(C, B)
unstack(C, B)
stack(A, C)
unstack(A, C)
stack(B, C)
unstack(B, C)
stack(B, A)
stack(B, A)
stack(C, A)
unstack(C, A)
CLEAR(A) ON(A, B) ON(B, C) ONTABLE(C) HANDEMPTY
CLEAR(C) ON(C, B) ON(B, A) ONTABLE(A) HANDEMPTY
CLEAR(A) ON(A, C) ON(C, B) ONTABLE(B) HANDEMPTY
CLEAR(B) ON(B, C) ON(C, A) ONTABLE(A) HANDEMPTY
CLEAR(B) ON(B, A) ON(A, C) ONTABLE(C) HANDEMPTY
CLEAR(C) ON(C, A) ON(A, B) ONTABLE(B) HANDEMPTY

Sussman Anomaly

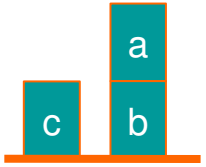
Actually discovered by Allen Brown

Initial state:
clear(c),
on(c, a)
ontable(a),
clear(b),
ontable(b),
handempty



Goal:
on(a, b)
on(b, c)

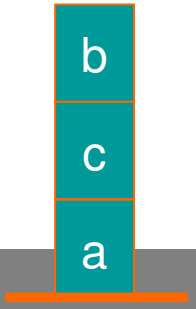
unstack(c, a)
putdown(c)
pickup(a)
stack(a, b)



Achieving on(a, b)

Achieving on(b, c)

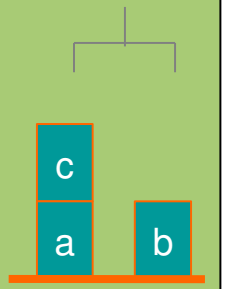
pickup(b)
stack(b, c)



UNIVERSITÄT DES SAARLANDES

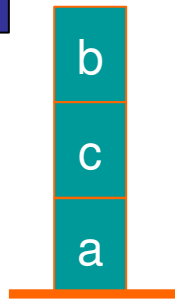
Sussman Anomaly: Undoing a Goal !

Initial state:
clear(c),
on(c, a)
ontable(a),
clear(b),
ontable(b),
handempty

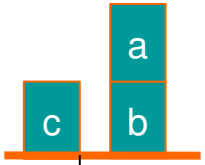


Goal:
on(a, b)
on(b, c)

pickup(b)
stack(b, c)



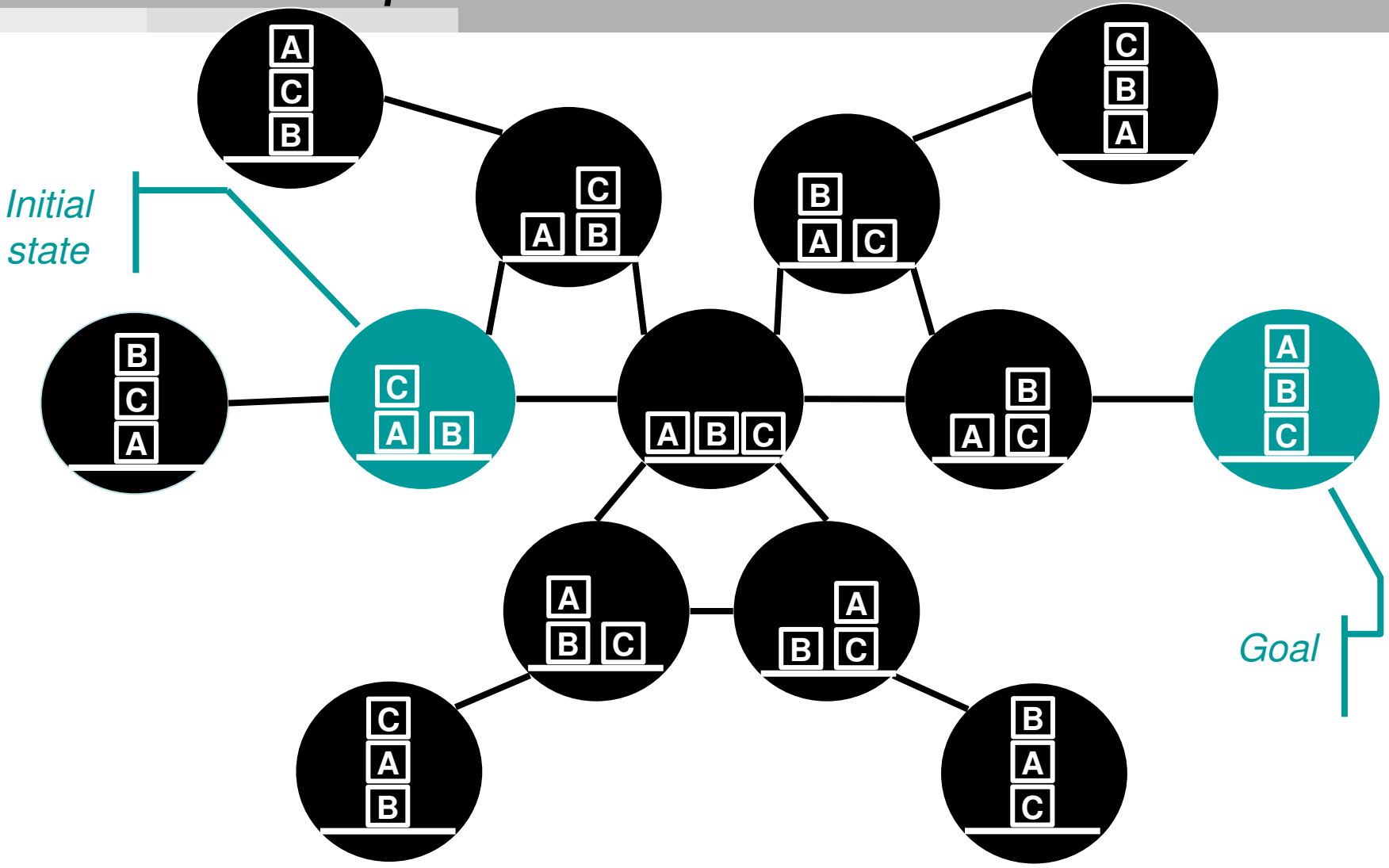
unstack(b, c)
putdown(b)
unstack(c, a)
putdown(c)
pickup(a)
stack(a, b)



Back in the Loop !!!

UNIVERSITÄT DES SAARLANDES

State-Space Search: ***State-space planning is a search in the space of states***



DISCUSSION: State-based Planning

- Partial Order of Plan Operators would be better
- Totally ordered plan cannot take advantage of problem decomposition
- Plan Space instead of State Space
- Partial Order Planning: POP-Planner

DISCUSSION: State-based Planning

Deleted-Condition Interaction:

- one of th
  previously

Deleted –Co

- Need a w

unstack(c, a)

not so “impo

Hence: Planning in the *Planspace!*

POP: Partial Order Planning