1

The Blocks World

- The blocks world
  - Representation
  - Actions
  - Precondition/Add/Delete lists
- Planning
- Means-ends reasoning
- Examples
- Sussman's Anomaly

2

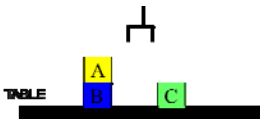
Questions

- How do we *represent*. . .
  - goal to be achieved
  - state of environment
  - actions available to agent
  - plan itself

3

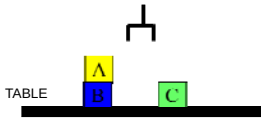
Blocks World

- We'll illustrate the techniques with reference to the blocks world
- This world contains
  - a robot arm with gripper,
  - 3 blocks (A, B and C) of equal size,
  - a table-top.



4

Blocks World



Some domain constraints:

- Only one block can be directly on top of another block
- Any number of blocks can be on the table
- The hand can only hold one block

5

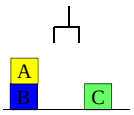
Ontology

- To represent this environment, we need an Ontology
- **On(x,y)** *means block x* is on top of *block y*
- **OnTable(x)** --- *block x* is on the table
- **Clear(x)** --- nothing is on top of *block x*
- **Holding(x)** --- *robot* arm is holding *block x*
- **ArmEmpty()** --- *robot* arm/hand is not holding anything (block in this world)

6

State Representation = Environment

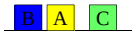
- A representation of one state of the blocks world. The state in the figure is:
  - *Clear(A)*
  - *Clear(C)*
  - *On(A,B)*
  - *OnTable(B)*
  - *OnTable(C)*
  - *ArmEmpty()*



- Use the *closed world assumption*: *anything not stated is assumed to be false*

Goal Representation

- A **goal** is represented as a set of formulae
- Here is a goal:
- *OnTable(A)*
- *OnTable(B)*
- *OnTable(C)*



Actions

Represented using a technique that was developed in the STRIPS planner

Each action has:

- a **name** which may have arguments;
- a **pre-condition list** --- a list of facts which must be true for action to be executed;
- a **delete list** --- a list of facts that are no longer true after action is performed;
- an **add list** --- a list of facts made true by executing the action.

Each of the facts may contain **variables**

Action/Operator Representation

- Basic operations
  - stack(X,Y): put block X on block Y
  - unstack(X,Y): remove block X from block Y
  - pickup(X): pickup block X from the table
  - putdown(X): put block X on the table
- Each operator is represented by facts that describe the state of the world before and changes to the world after an action is performed.
  - a list of **preconditions**
  - a list of new **facts to be added** (add-effects)
  - a list of **facts to be removed** (delete-effects)
  - optionally, a set of (simple) variable **constraints**

Precondition/Delete/Add Lists

- Preconditions
  - P1 … Pi
- Additions
  - A1 … Ak
- Deletions
  - D1 … Dj
- Meaning:
- All P must be true before an action is performed (otherwise it can't be accomplished)
- After the action, all A are added to the agent's memory/state
- After the action, all D are removed from the agent's memory/state
- Subject to constraints imposed on the state of the world
  - e.g. a block can't be stacked on top of itself!!

Stack Operator

- The **stack** action occurs when the robot arm places the object it is holding [x] on top of another object [*y*]
- Form: *Stack(x,y)*
- Pre: $Clear(y) \wedge Holding(x)$
- Add: $ArmEmpty \wedge On(x,y) \wedge Clear(x)$
- Del: $Clear(y) \wedge Holding(x)$
- Constraints: $(x \neq y), x \neq Table, y \neq Table$

Unstack Operator

- The **unstack** action occurs when the robot arm picks up an object *x* from on top of another object *y*.
- Form: *UnStack(x,y)*
- Pre: $On(x,y) \wedge Clear(x) \wedge ArmEmpty()$
- Add: $Holding(x) \wedge Clear(y)$
- Del: $On(x,y) \wedge Clear(x) \wedge ArmEmpty()$
- Constraints: $x \neq y, x \neq Table, y \neq Table$

13

Pickup Operator

- The **pickup** action occurs when the arm picks up an object (block) from the table
- Form: *Pickup(x)*
- Pre: $OnTable(x) \wedge Clear(x) \wedge ArmEmpty()$
- Add: *Holding(x)*
- Del: $OnTable(x) \wedge Clear(x) \wedge ArmEmpty()$
- Constraints: $x \neq table$

14

Putdown Operator

- The **putdown** action occurs when the arm places the object *x* onto the table
- Form: *PutDown(x)*
- Pre: *Holding(x)*
- Add: $OnTable(x) \wedge ArmEmpty \wedge Clear(x)$
- Del: *Holding(x)*
- Constraints: $x \neq table$

15

Planning and Agents

- Since the early 1970s, the AI planning community has been closely concerned with the design of artificial agents
- Planning is essentially *automatic programming*: the design of a course of action that will achieve some desired goal
- Within the symbolic AI community, it has long been assumed that some form of AI planning system will be a central component of any artificial agent
- Building largely on the early work of Fikes & Nilsson, many planning algorithms have been proposed, and the theory of planning has been well-developed

16

What is a Plan?

- A sequence (list) of actions, with variables replaced by constants (specific objects in the environment)

17

Planner

18

Means-Ends Reasoning

- Idea is to give an *agent*:
  - representation of goal/intention to achieve;
  - representation of actions it can perform; and
  - representation of the environment;
- Then have the agent **generate a plan to achieve the goal.**
- The plan is generated entirely by the planning system, without human intervention.

19

STRIPS Planning

- STRIPS maintains two additional data structures:
  - **State List** - all currently true predicates.
  - **Goal Stack** - a push down stack of goals to be solved, with current goal on top of stack.
- *If the current goal is not satisfied* by present state,

  Find goal in the *add list of an operator*,

  and

  push *operator and preconditions list on stack.* (=Subgoals)
- When a *current goal is satisfied*, POP it from stack.
- When an *operator is on top of the stack*,
  - record the application of that operator – update the plan sequence

    and
  - use the operator's add and delete lists to update the current state.

20

Planning in STRIPS

- Uses means-ends reasoning (actions = means, goals = ends)
- States of the world and goals are represented as a set/list of predicates that are true (e.g. on(x,y) ..)

1. The current state is initialized to the start state
2. The goal is placed on the goal stack
3. Loop through the following steps to produce a plan

   (next slide)

21

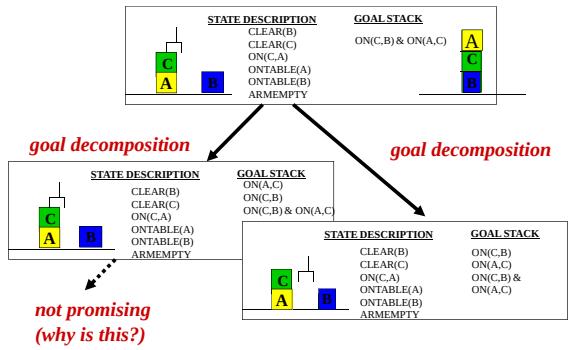
Reasoning Loop

If the top item on the goal stack is:

- empty (the goal stack is empty), return the actions executed – they form the plan to achieve the goal
- a goal, and it is satisfied in the current state, remove it from the stack (no replacement necessary)
- a complex goal, break it into subgoals, placing all subgoals on the goal stack (the original goal is pushed down in the goal stack)
- a predicate, find an action that will make it true, then place that action (with variables bound appropriately) and its preconditions on the goal stack (preconditions first)
- an action and its preconditions are satisfied, perform the action, updating the world state using the delete and add lists of the action (if the pre-conditions are not satisfied, add them to the goal list without removing the action). Add this action to the partial plan

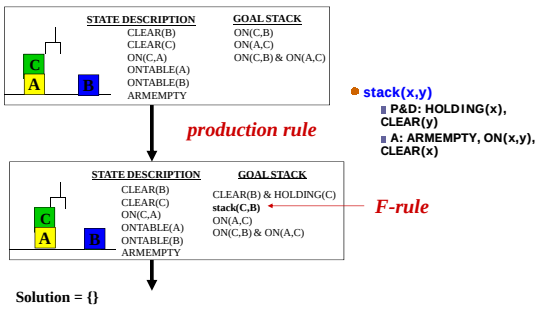
22

STRIPS in action



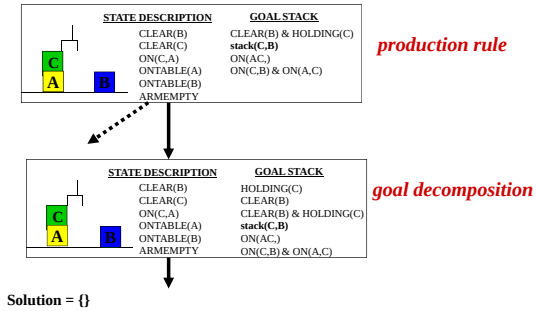
23

STRIPS in action



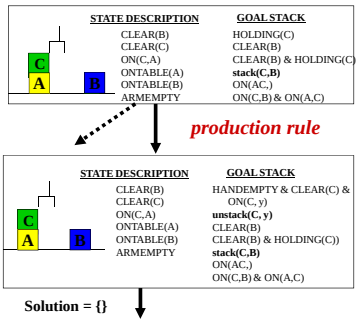
24

STRIPS in action



25

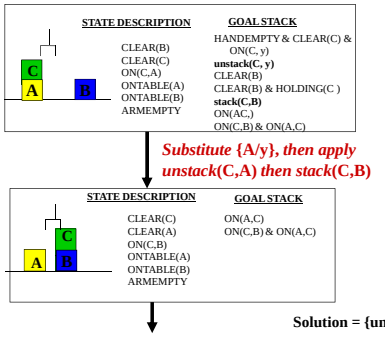
STRIPS in action



- unstack(x,y)
  - P&D: HANDEMPTY, CLEAR(x), ON(x,y)
  - A: HOLDING(x), CLEAR(y)

26

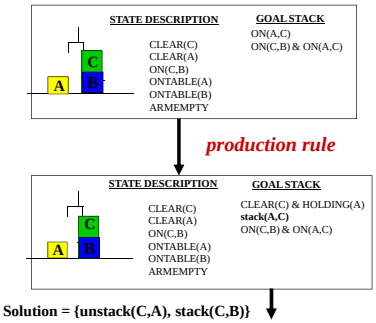
STRIPS in action



- unstack(x,y)
  - P&D: ARMEMPTY, CLEAR(x), ON(x,y)
  - A: HOLDING(x), CLEAR(y)
- stack(x,y)
  - P&D: HOLDING(x), CLEAR(y)
  - A: ARMEMPTY, ON(x,y), CLEAR(x)

27

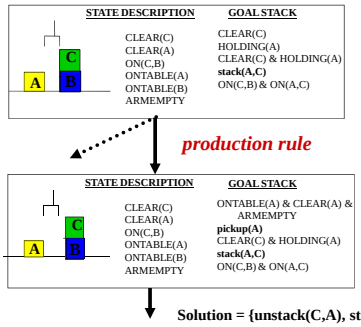
STRIPS in action



- stack(x,y)
  - P&D: HOLDING(x), CLEAR(y)
  - A: ARMEMPTY, ON(x,y), CLEAR(x)

28

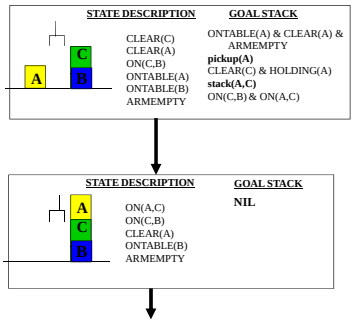
STRIPS in action



- pickup(x)
  - P&D: ONTABLE(x), CLEAR(x), ARMEMPTY
  - A: HOLDING(x)

29

STRIPS in action



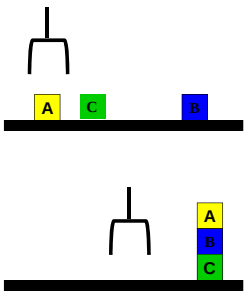
- pickup(x)
  - P&D: ONTABLE(x), CLEAR(x), HANDEMPTY
  - A: HOLDING(x)
- stack(x,y)
  - P&D: HOLDING(x), CLEAR(y)
  - A: HANDEMPTY, ON(x,y), CLEAR(x)

30

Typical BW Planning Problem

Initial state:
- clear(a)
- clear(b)
- clear(c)
- ontable(a)
- ontable(b)
- ontable(c)
- handempty

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



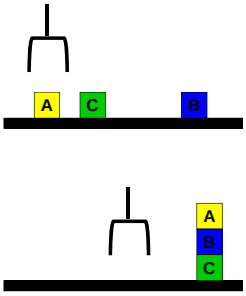
A plan:
- pickup(b)
- stack(b,c)
- pickup(a)
- stack(a,b)

31

Another BW Planning Problem

Initial state:
- clear(a)
- clear(b)
- clear(c)
- ontable(a)
- ontable(b)
- ontable(c)
- handempty

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



A plan:
- pickup(a)
- stack(a,b)
- unstack(a,b)
- putdown(a)
- pickup(b)
- stack(b,c)
- pickup(a)
- stack(a,b)

32

Goal Interaction

- Simple planning algorithms assume that the goals to be achieved are independent
  - Each can be solved separately and then the solutions concatenated
- This planning problem, called the "Sussman Anomaly," is the classic example of the goal interaction problem:
  - Solving on(A,B) first (by doing unstack(C,A), stack(A,B) will be undone when solving the second goal on(B,C) (by doing unstack(A,B), stack(B,C)).
  - Solving on(B,C) first will be undone when solving on(A,B)
- Classic STRIPS could not handle this, although minor modifications can get it to do simple cases

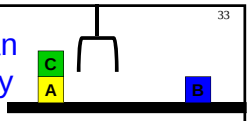
Initial state

Goal state

Achieve on(a,b) via stack(a,b) with preconds: [holding(a),clear(b)]
|Achieve holding(a) via pickup(a) with preconds: [ontable(a),clear(a),handempty]
||Achieve clear(a) via unstack(_1584,a) with preconds: [on(_1584,a),clear(_1584),handempty]
||Applying unstack(c,a)
||Achieve handempty via putdown(_2691) with preconds: [holding(_2691)]
||Applying putdown(c)
|Applying pickup(a)
Applying stack(a,b)
Achieve on(b,c) via stack(b,c) with preconds: [holding(b),clear(c)]
|Achieve holding(b) via pickup(b) with preconds: [ontable(b),clear(b),handempty]
||Achieve clear(b) via unstack(_5625,b) with preconds: [on(_5625,b),clear(_5625),handempty]
||Applying unstack(a,b)
||Achieve handempty via putdown(_6648) with preconds: [holding(_6648)]
||Applying putdown(a)
|Applying pickup(b)
Applying stack(b,c)
Achieve on(a,b) via stack(a,b) with preconds: [holding(a),clear(b)]
|Achieve holding(a) via pickup(a) with preconds: [ontable(a),clear(a),handempty]
|Applying pickup(a)
Applying stack(a,b)

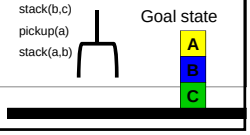
33

Sussman Anomaly



From [clear(b),clear(c), ontable(a),ontable(b),on(c,a),handempty]
To [on(a,b),on(b,c),ontable(c)]
Do: unstack(c,a)
- putdown(c)
- pickup(a)
- stack(a,b)
- unstack(a,b)
- putdown(a)
- pickup(b)
- stack(b,c)
- pickup(a)
- stack(a,b)

Goal state



34

Questions