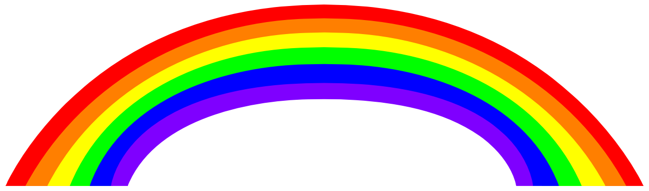
Artificial Intelligence
Planning

Lecture 9

(27 October, 1999)

Tralvex (Rex) Yeap MAAI MSCS
University of London

Content: Planning



- Quick Review of Lecture 9
- Introduction to Planning
- Examples of Planning Systems
- Blocks World
- Assumptions of the "Standard" AI Planning Paradigm
- STRIPS - Linear Planner
- STRIPS Example
- State Space Searching
  - Progression Planners
  - Regression Planners
- Plan Space Searching
- Partial Ordered Planning
  - Introduction
  - An Example
  - Interpretation
- Partially ordered plans vs. Non-linear planning
- Shortcomings of AI Planning in General
- Students' Mini Research Presentation by Group D
- Class Activity: Real-world Papers Reading
- What's in Store for Lecture 10

Quick Review on Lecture 8

- First Order Logic BNF Grammer
- Class Workout 1: Express them in LFOPC and draw the diagrams (with solutions)
- Class Workout 2: Prove that Colonel West is a criminal
- Logic for Commonsense Reasoning
- Introduction to Non-monotonic Reasoning
- Representation using Semantic Nets
- Semantic Nets Representation
- Semantic Nets Inference Mechanism
- Representation using Frames (Box on Table Example)
- Other kinds of Logic
- Students' Mini Research Presentation by Group C

Introduction to Planning

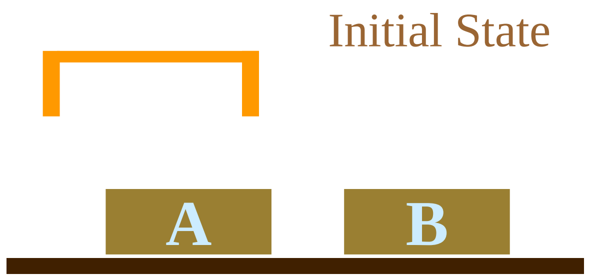
- Plan: a **sequence of steps** to achieve a goal.
- Problem solving agent knows: **Actions**, **states**, **goals** and **plans**.
- Planning is a **special case** of problem solving: reach a **state** satisfying the **requirements** from the current state using **available actions**.

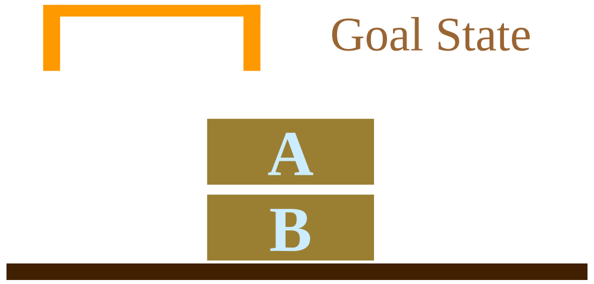
Examples of Planning Systems

- Spacecraft assembly, integration and verification
- Job shop scheduling
- Space mission scheduling
- Building construction
- Operations on a flight deck of an aircraft carrier
- For demos: blocks world

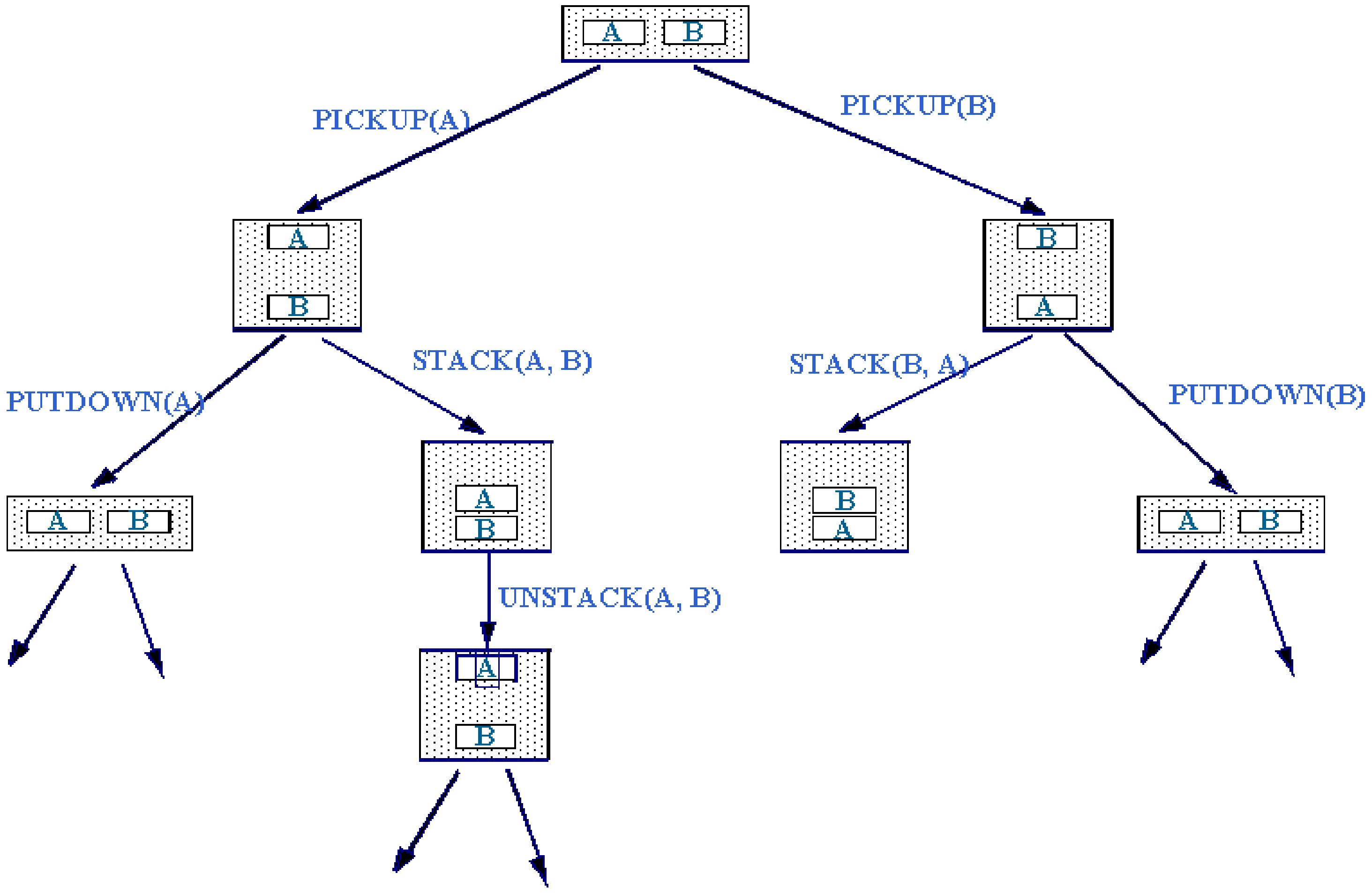
Blocks World

- PickUp(X)
  - X on table, hand empty, X free
- PutDown(X)
  - X in hand
- Stack(X,Y)
  - X in hand, y free
- Unstack(X,Y)
  - X free, X on Y, hand free





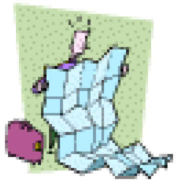
Blocks World (cont)



Assumptions of the "Standard" AI Planning Paradigm

- There is a **single causal agent** and this agent is the planner.
- The planner is given a **well-defined goal** which remains **fixed** over the course of planning.
- The planner is assumed to have **functionally complete** and **accurate knowledge** of the starting situation.
- The planner is assumed to **possess** the **knowledge** required to accurately model the world.
- The planner is assumed to **possess** the **resources** (time and memory) required to use this model to reason about the possible worlds associated with different courses of action that might be pursued.

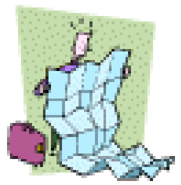
STRIPS - Linear Planner



- First planner developed by SRI, stands for STanford Research Institute Problem Solver.
- In STRIPS notation, a model of the world is just a **list of variables free atomic propositions** that hold in the world.
- **Operators involving variables** are called **operator schemas**.
- The following expresses an initial state in the block world:

  <on(a,t), on(b,a),clear(b),on(c,t),clear(c))>
- It is assumed that **anything not mentioned** in the description of the initial state of the world is **false**.

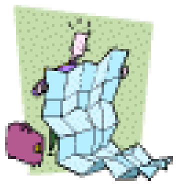
STRIPS



- The **description** of the goal state is again a **list of atomic proposition** where all variables are interpreted existentially.

- The goal state of plan, for example, will be given by such a description (if we want an apple we usually do not refer to a particular apple). An example goal state in the block world is:
  <on(X,c), on(c,t)>
  This means that some block should be on c, which is itself directly on the table

STRIPS (cont)



- The main element of the language is the operator description, which has **three parts**:

  (1) the **action** name, which may be parametrized

  (2) the **precondition**, which is a conjunction of positive literals

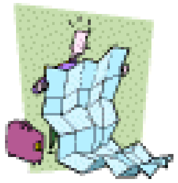
  (3) the **effect**, which is a conjunction of positive and/or negative literals

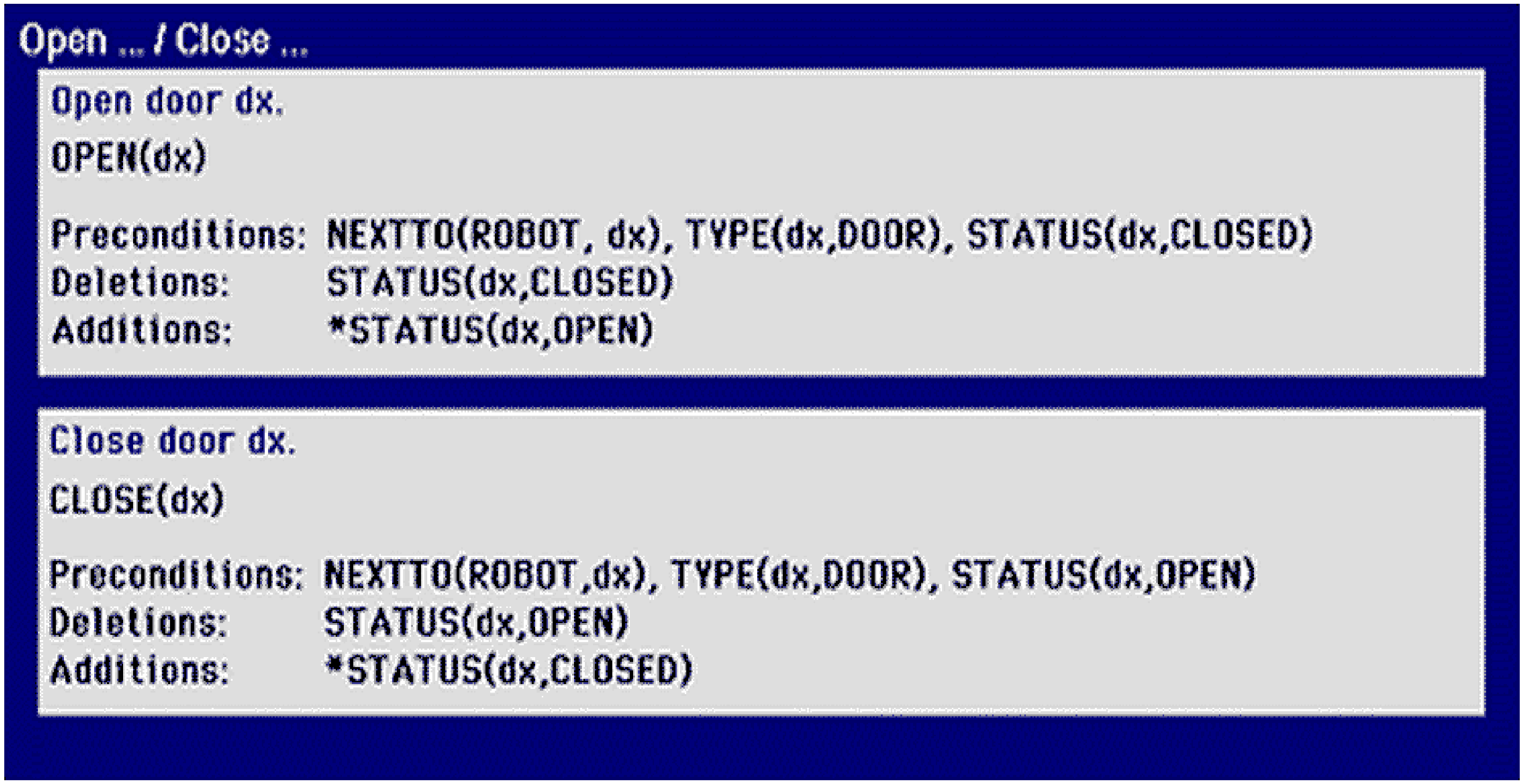
- The Preconditions consist of a **conjunctive logical expression** which is intended to describe the conditions that must be true in order to apply the operator.

- The **positive or additions** consist of a set of expressions that must be **added** to a model of the situation if the operator is applied.

- The **negative or deletions** consist of a set of expressions that must be **deleted** from a model of a situation if the operator is applied.

STRIPS (cont)





Open ... / Close ...

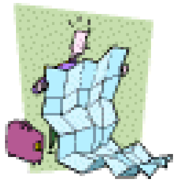
Open door dx.

```
OPEN(dx)

Preconditions:  NEXTTO(ROBOT, dx), TYPE(dx,DOOR), STATUS(dx,CLOSED)
Deletions:      STATUS(dx,CLOSED)
Additions:      *STATUS(dx,OPEN)
```

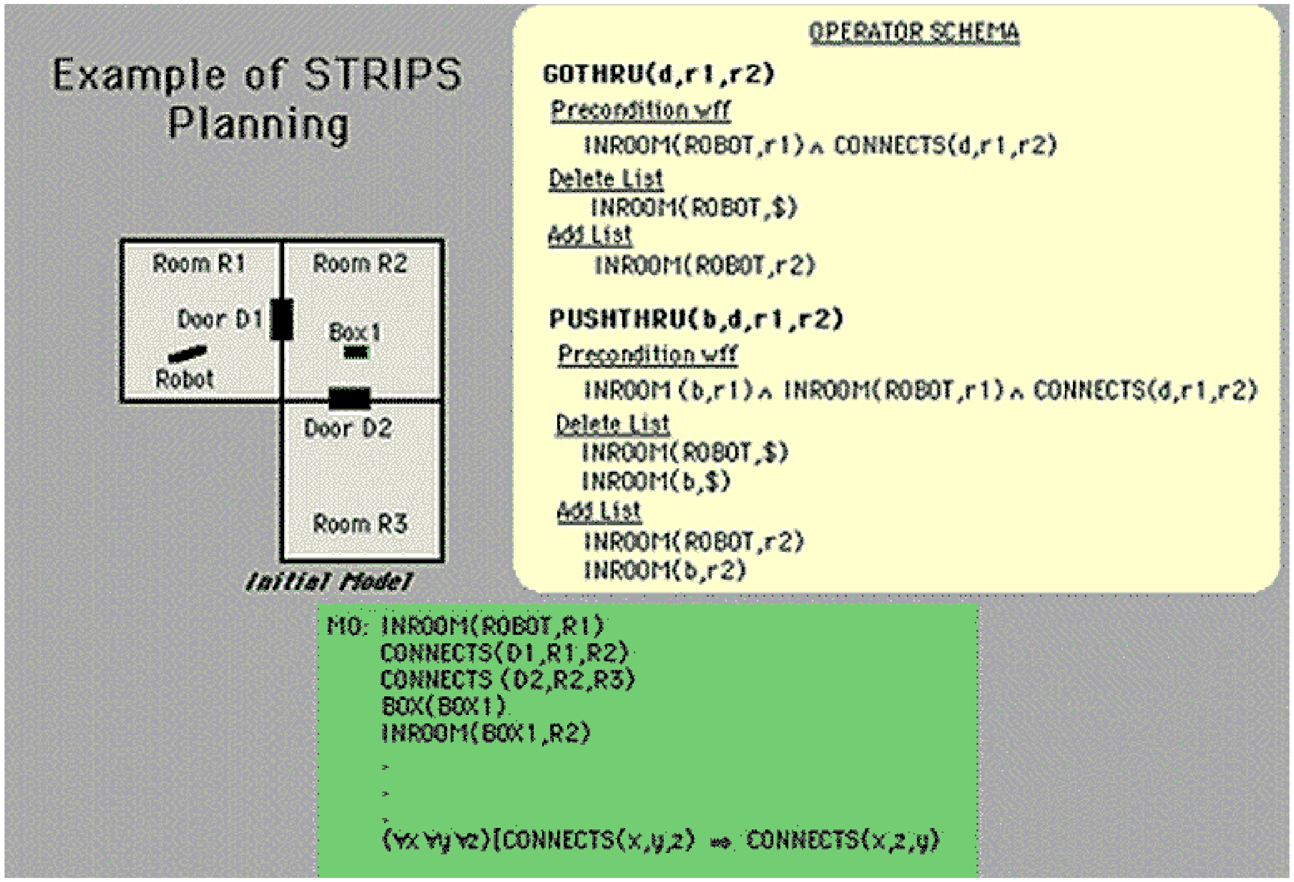
Close door dx.

```
CLOSE(dx)

Preconditions:  NEXTTO(ROBOT,dx), TYPE(dx,DOOR), STATUS(dx,OPEN)
Deletions:      STATUS(dx,OPEN)
Additions:      *STATUS(dx,CLOSED)
```

Figure 10.1. STRIPS representation for (1) opening a door (2) closing a door.



STRIPS Example



OPERATOR SCHEMA

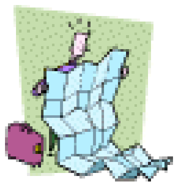
GOTHRU(d,r1,r2)
Precondition wff
INROOM(ROBOT,r1) ∧ CONNECTS(d,r1,r2)
Delete List
INROOM(ROBOT,$)
Add List
INROOM(ROBOT,r2)

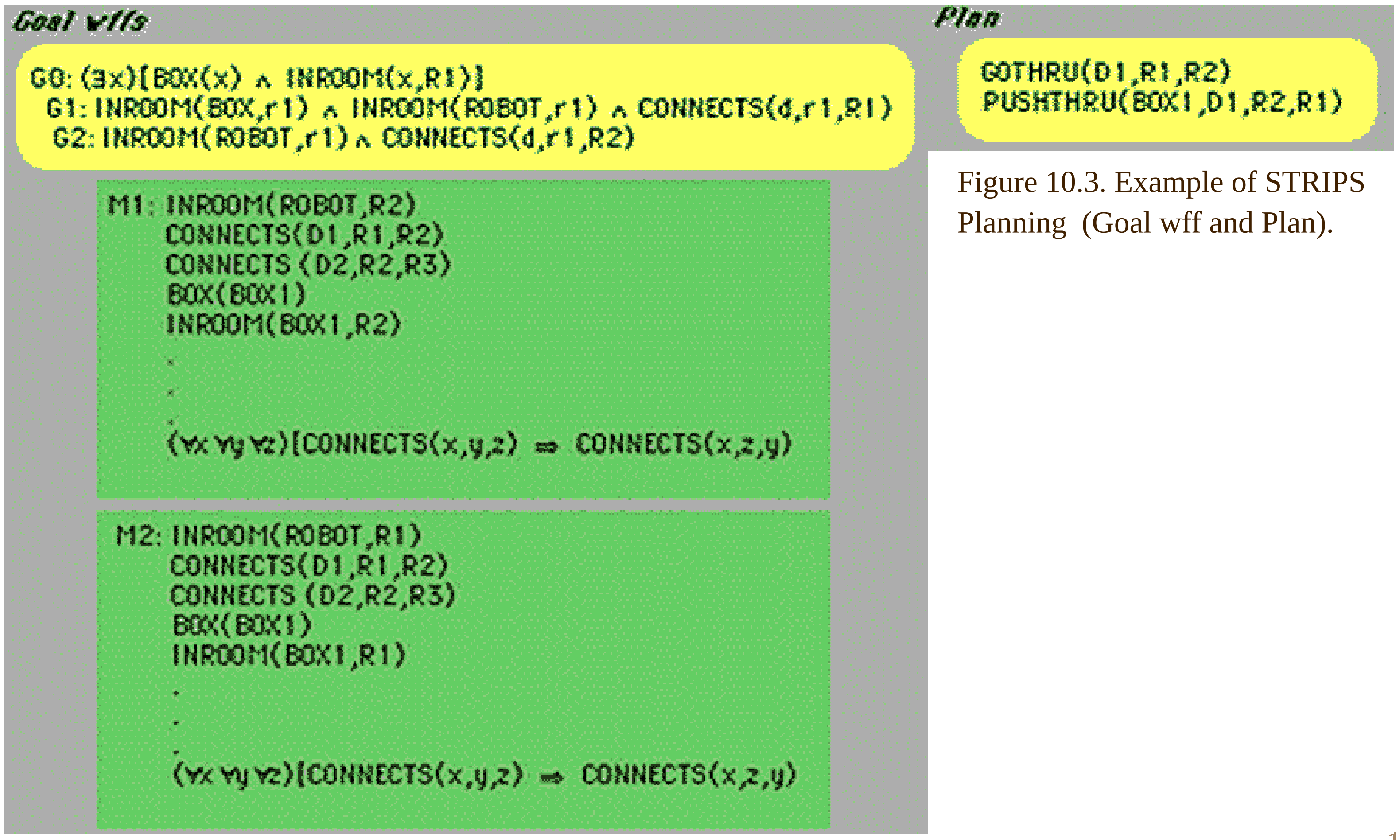
PUSHTHRU(b,d,r1,r2)
Precondition wff
INROOM (b,r1) ∧ INROOM(ROBOT,r1) ∧ CONNECTS(d,r1,r2)
Delete List
INROOM(ROBOT,$)
INROOM(b,$)
Add List
INROOM(ROBOT,r2)
INROOM(b,r2)

```
M0: INROOM(ROBOT,R1)
    CONNECTS(D1,R1,R2)
    CONNECTS (D2,R2,R3)
    BOX(BOX1)
    INROOM(BOX1,R2)
    .
    .
    .
    (∀x∀y∀z)[CONNECTS(x,y,z) ⇒ CONNECTS(x,z,y)
```

Figure 10.2. Example of STRIPS Planning (Operator Schema & Initial Model).

STRIPS Example (cont)





Goal wffs

G0: (∃x)[BOX(x) ∧ INROOM(x,R1)]
G1: INROOM(BOX,r1) ∧ INROOM(ROBOT,r1) ∧ CONNECTS(d,r1,R1)
G2: INROOM(ROBOT,r1) ∧ CONNECTS(d,r1,R2)

```
M1: INROOM(ROBOT,R2)
    CONNECTS(D1,R1,R2)
    CONNECTS (D2,R2,R3)
    BOX(BOX1)
    INROOM(BOX1,R2)
    .
    .
    .
    (∀x ∀y ∀z)[CONNECTS(x,y,z) ⇒ CONNECTS(x,z,y)
```

```
M2: INROOM(ROBOT,R1)
    CONNECTS(D1,R1,R2)
    CONNECTS (D2,R2,R3)
    BOX(BOX1)
    INROOM(BOX1,R1)
    .
    .
    .
    (∀x ∀y ∀z)[CONNECTS(x,y,z) ⇒ CONNECTS(x,z,y)
```

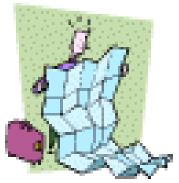
Plan

```
GOTHRU(D1,R1,R2)
PUSHTHRU(BOX1,D1,R2,R1)
```

Figure 10.3. Example of STRIPS Planning (Goal wff and Plan).

State Space Searching

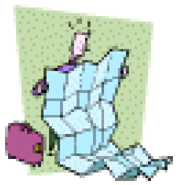
Progression Planners



- Search **top-down** from initial state to the goal state.
- This algorithm will build a path from the **initial state** to the **goal**.
- The algorithm also **keep a record** of the plan it has built at any stage to the current state.

State Space Searching

Progression Planners (cont)



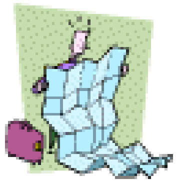
- Progression planners can use **any of the search methods**, both blind and heuristic incline.

- A depth-first search algorithm is summarised below:

  1. If the current state *S* satisfies the goal then return the path.
  2. Else,
     (a) try and pick an appropriate action *A* whose precondition is satisfied by the current world state.
     (b) if there is no such action, then backtrack.
     (c) else, move to the state in the search space *S'* that would result from performing that action in the current state *S*,
     and then find a path (plan) *P'* that goes from that new state to the goal.
     Returning the complete path *P* from *S* to the goal (where *P* = [*A*|*P'*], the list of actions consisting of *A* followed by *P*).

State Space Searching

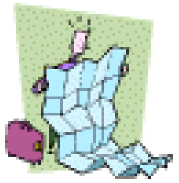
Regression Planners



- Search **bottom-up** from the goal state to the initial state.
- This algorithm build a path from the **goal** to the **initial state**.
- The algorithm also **keep a record** of the plan it has built at any stage to the current state.

State Space Searching

Regression Planners (cont)

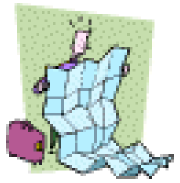


- Similarly regression planners can use **any of the search methods**.

- A depth-first search algorithm is summarised below:

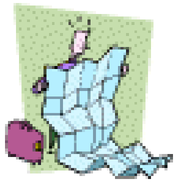
1. If the current state *S* satisfies the goal then return the path.
2. Else,
   (a) try and pick an appropriate action *A* whose effect is satisfied by the current world state.
   (b) if there is no such action, then backtrack.
   (c) else, move to an appropriate state in the search space *S'* that would satisfy the preconditions of the action *A*, and that would result in the state *S*, if *A* was performed, and then recursively find a path (plan) *P'* that goes from that new state to the initial state.
   Returning the complete path *P* from *S* to the initial state (where *P* = [*A*|*P'*], the list of actions consisting of *A* followed by *P*).

Plan Space Searching



- An alternative way of viewing the planning problem is to see it as a **search through possible plans**.
- The main motivation for plan space searching is to **avoid back-tracking** by looking at the goals in an order different from execution order.
- Search space consisting of **states** of the world are **linked by actions**.

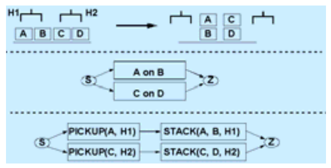
Plan Space Searching (cont)



- Plan space is a **collection** of **partially specified plans** linked by operators that refine a plan into a more detailed one.
- **The initial plan**, is some **unspecified actions** that takes the initial state into the goal state.
- The goal will be **fully specified plan** (or plans) that performs the desired function.

Partial Ordered Planning

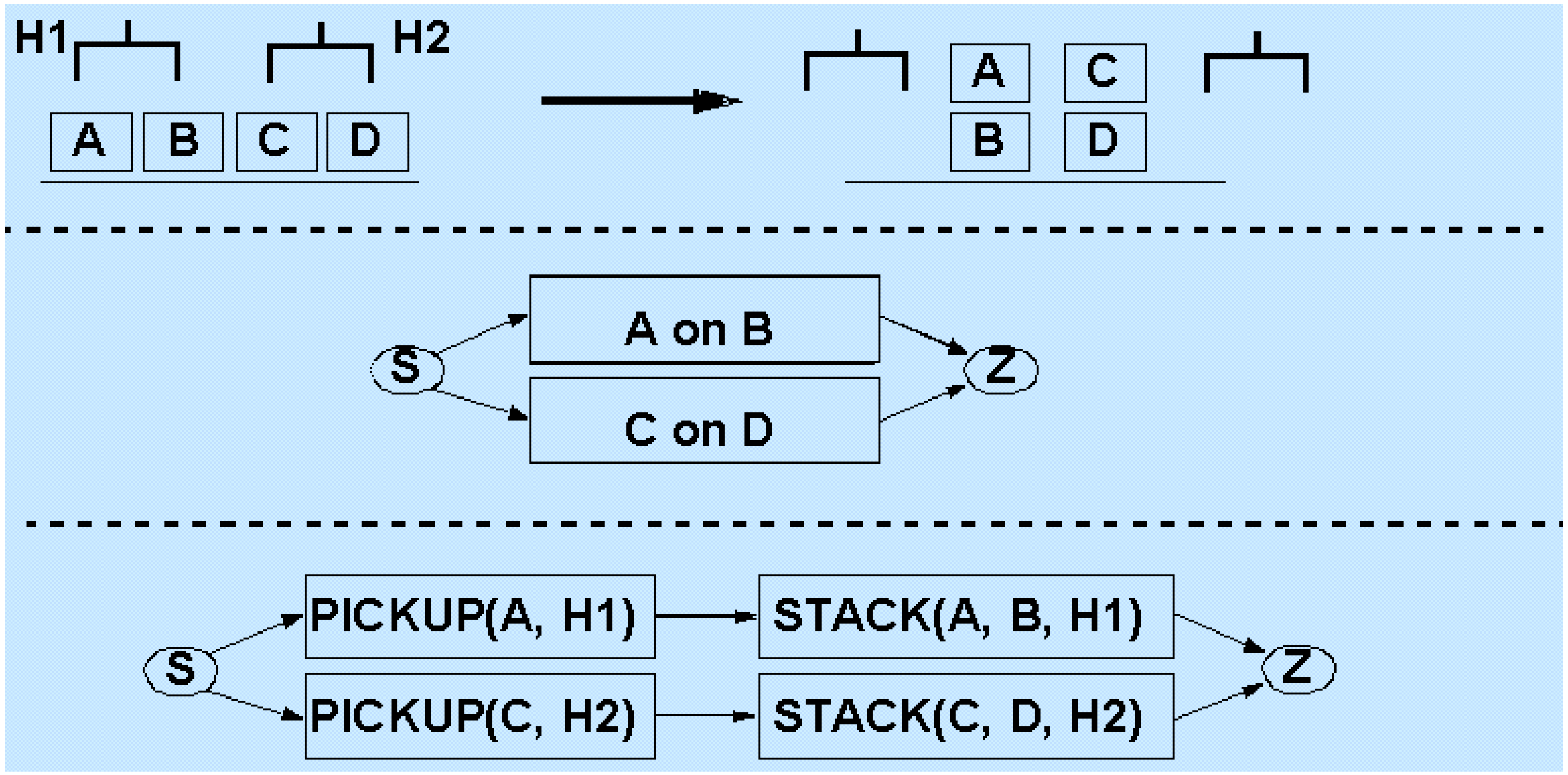
Introduction



- A partially ordered plan is a general representation of plans.
- Idea:
  - Working parallel on several sub-goals.
  - Ordering of goals based on interactions.
- Underlying assumption:
  - Not many interactions.
- Partially ordered plan = directed graph (AND).

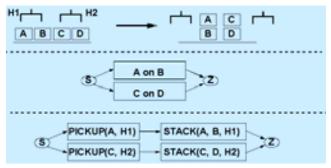
Partial Ordered Planning

An Example



Partial Ordered Planning

Interpretation

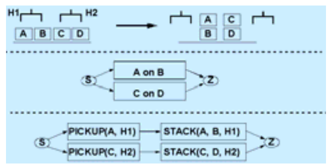


- An ordered pair **P = (O,<)** is a **plan** :⇔ **O** is a set of **nodes**, < is a strict **partial order** on O with definite smallest element *Start(P)* and definite largest element *Goal(P)*

- A partially ordered plan can be executed in **any total order** that is **compatible** with the **partial order**.
  - PICKUP(A, H1), PICKUP(C, H2), Stack(A,B,H1), Stack(C,D,H2)
  - PICKUP(A, H1), Stack(A,B,H1), PICKUP(C, H2), Stack(C,D,H2)
  - not ok: PICKUP(A, H1), Stack(A,B,H1), Stack(C,D,H2), PICKUP(C, H2)

Partial Ordered Planning

Interpretation (cont)



- **Stronger interpretation** (less execution possibilities): Parallel branches can be ordered only in total.

- **Weaker interpretation** (more execution possibilities): Parallel execution allowed.

Partially ordered *plans* vs. Non-linear *planning*

- Representation of the plan: partially ordered plan
- Inserting parts of plans at an arbitrary location: non linear planning

Shortcomings of AI Planning in General

- Not every actions can be described with STRIPS-like operators:
  - money transfer: new balance is a function of the old
  - alternative post-conditions
    - PAINTBLACK(x)
      precondition: x is white
      (Why not blue? to know what to delete!)

- No complete knowledge about the world.

Shortcomings of AI Planning in General (cont)

- The world is not stable
  - ➢ Re-planning must be supported
- Goals are not clearly defined
- Nobody plans the solution of everyday tasks
- Humans learn

Class Activity: Real-world Paper Reading

Paper 1. “Why Real-World Planning is Difficult: A Tale of Two Applications”

- **Introduction**
- **The MVP and LMCOA Applications**
  - MVP: Automated VICAR Image Processing
  - LMCOA and the Deep Space Network
- **Representation Issues**
  - Representation Issues in MVP
  - Representation Issues in LMCOA
- **Operational Contexts**
  - Operational Contexts and MVP
  - Operational Context and LMCOA
- **Knowledge Acquisition and Knowledge Base Maintenance**
  - Knowledge Acquisition and Maintenance in MVP
  - Knowledge Acquisition and Maintenance in LMCOA
- **Summary**

Class Activity: Real-world Paper Reading

Paper 2. “Planning to Gather Information”

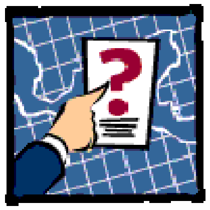
- **Introduction**
  - Assumptions
  - A Simple Example
  - Context
  - Overview
- **Representing the World, Sites and Queries**
  - Representing Information-Producing Sites
  - Representing an Information Gathering Query
- **Plan & Solution**
  - Plans
  - Solution to Query
- **Planning to Gather Information**
  - The Example, Revisited
  - Finding Solutions from Sequences
  - The Example, Continued
  - Transformation Based on Equality Mappings
  - Redundant Solutions
  - Formal Properties
- **Reducing Search**
  - Pruning Plans with Duplicate Operator Instances
  - Pruning Shuffled Sequences
  - Experimental Validation
- **Finding Simplest Plans**
- **Related Work**
- **Conclusion**

Class Activity: Real-world Paper Reading

Paper 3. “Recent Advances in AI Planning”

- **Introduction**
  - Preliminaries
  - Available Implementations
- **Graphplan & Descendant**
  - Expanding the Planning Graph
  - Solution Extraction
  - Optimizations
  - Handling Expressive Action Languages
- **Compilation of Planning to SAT**
  - The Space of Encoding
  - Optimizations
  - SAT Solvers
- **Interleaved Planning & Execution Monitoring**
- **Discussion**

What's in Store for Lecture 10



- Introduction to NLP
- NL and Computer Language
- Motivations for NLP
- NLP History
- Major NLP Accomplishments
- Real World NLP Applications
  - MT: Deluxe Universal Translator
  - IR: Buzzcity
  - IR: Altavista Search Engine
  - IV: Cartia's Themescape
  - Autonomous interacting bots: Eliza's grand-daughter - Lisa
  - Grammer Checking Systems: MS Word Grammer Checker
- A Generic NL System Architecture
- Language and Knowledge
- Five Processing Stages in a NLP System
  - (1) Phonological Analysis
  - (2) Morphological Analysis
  - (3) Syntactic Analysis
  - (4) Semantic Analysis
  - (5) Pragmatic Analysis
- Class Activity: Real-world Paper Reading
- Students' Mini Research Presentation by Group E

End of Lecture 9

Good Night.