Title: Practical planning

AIMA: Chapter 12 (Sections 12.1-12.2)

Introduction to Artificial Intelligence

CSCE 476-876, Spring 2000

`www.cse.unl.edu/~cse476`

Berthe Y. Choueiry

`choueiry@cse.unl.edu, (402)472-5444`

Summary

- Situation space vs. plan space
- Progression vs. Regression
- Least commitment principle
- Ordering constraints and causal links
- Threat resolution: promotion, demotion
- POPlanner: algorithm

POPlanner algorithm

```
function POP(initial, goal, operators) returns plan

   plan ← MAKE-MINIMAL-PLAN(initial, goal)
   loop do
      if SOLUTION?(plan) then return plan
      S_need, c ← SELECT-SUBGOAL(plan)
      CHOOSE-OPERATOR(plan, operators, S_need, c)
      RESOLVE-THREATS(plan)
   end
```

```
function SELECT-SUBGOAL(plan) returns S_need, c

   pick a plan step S_need from STEPS(plan)
      with a precondition c that has not been achieved
   return S_need, c
```

```
procedure CHOOSE-OPERATOR(plan, operators, S_need, c)

   choose a step S_add from operators or STEPS(plan) that has c as an effect
   if there is no such step then fail
   add the causal link S_add --c--> S_need to LINKS(plan)
   add the ordering constraint S_add ≺ S_need to ORDERINGS(plan)
   if S_add is a newly added step from operators then
      add S_add to STEPS(plan)
      add Start ≺ S_add ≺ Finish to ORDERINGS(plan)
```

```
procedure RESOLVE-THREATS(plan)

   for each S_threat that threatens a link S_i --c--> S_j in LINKS(plan) do
      choose either
         Promotion: Add S_threat ≺ S_i to ORDERINGS(plan)
         Demotion: Add S_j ≺ S_threat to ORDERINGS(plan)
      if not CONSISTENT(plan) then fail
   end
```

→ Backtrack: when fail to resolve a threat or find operator

→ Select-subgoal is <u>not</u> subject to backtracking

Outline

- Strategies
- Types/strata
- General framework
- Practical Planners
- Hierarchical decomposition
  - extending the language
  - modifying the planner

Planning strategies:

- *Predictive*:
  goals and conditions are known beforehand
- *Dynamic* (on-line):
  plan actions as they are needed or conditions become available
- *Reactive*:
  update a predictive plan to accommodate unforeseen events

→ Most past research focused on predictive planners

→ Dynamic planners may miss opportunities and be wasteful of resources

→ Reactive planners are more credible for real-world settings

Planning *strata*: simplest model

In practice, predictive planning can be separated into 3 main strata:

1. **Planning**:

   Problem: decide which actions/tasks to undertake

   Input: initial state, goal state, operators

   Output: actions, ordering constraints, bindings

2. **Scheduling**:

   Problem: find execution times for tasks

   Input: actions, ordering constraints, durations

   Output: sequencing of actions and tasks

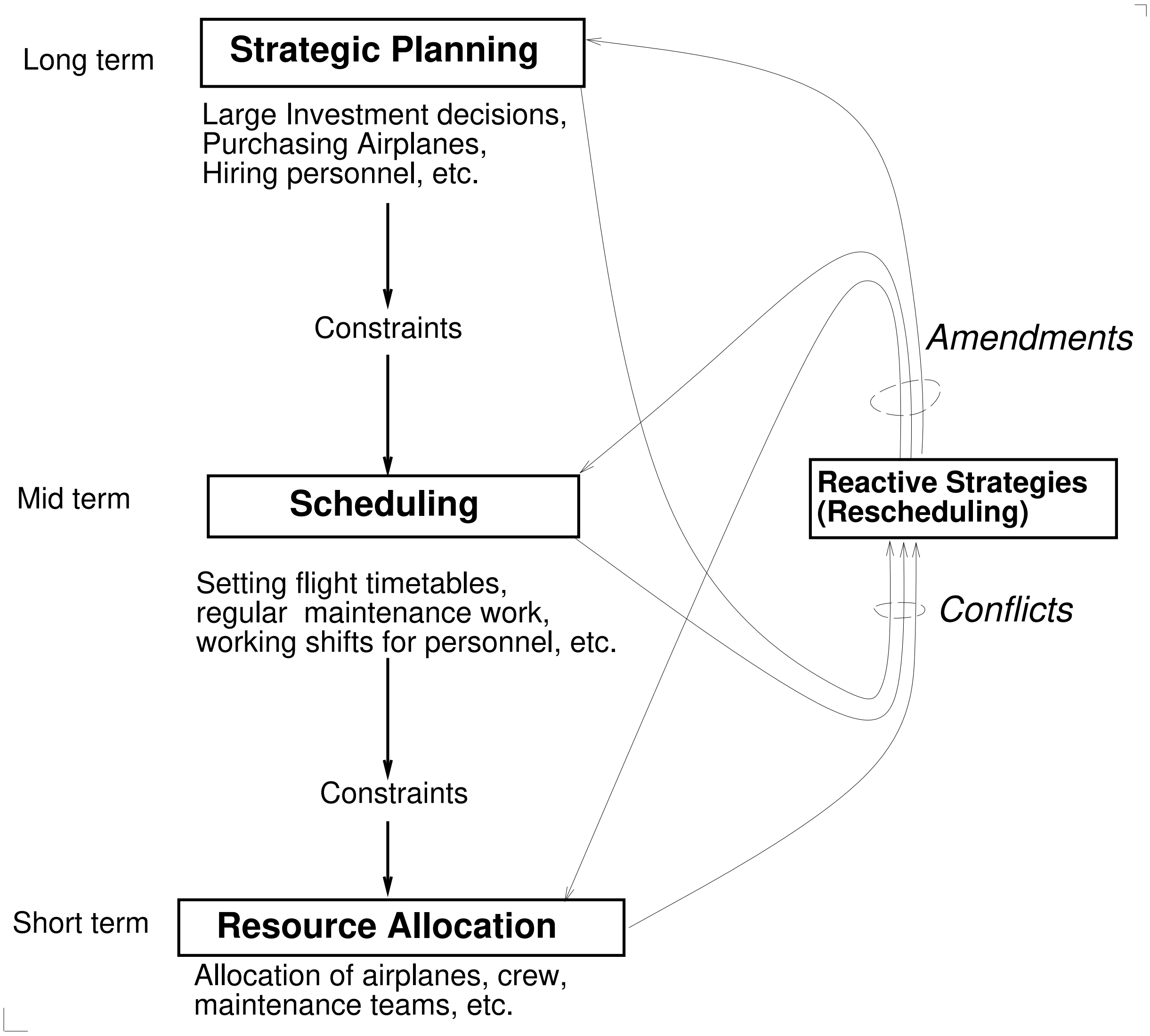
3. **Resource allocation**:

   Problem: determine which resources are allocated to which tasks

   Input: sequences of tasks, resource availability and capacities

   Output: assignment of resources to tasks

General framework



In practice, there exists no clear division among the strata

Practical Planners:

- Optimum-AIV (O-Plan)
- Job-shop scheduling (Tosca, Isis)
- Scheduling of space missions (HST: Spike, SPSS)
- Buildings, aircraft carriers, beer factories (SIPE)
- Resource allocation: personnel, gate in airports, aircraft for flights, etc.
- NASA Ames' PS in Remote Agent: `ic-www.arc.nasa.gov/ic/projects/remote-agent/index.html`

Practical Planners Optimum-AIV (I)

- ESA: Assembly, Integration and Verification of spacecraft
- Generates plans, monitors their execution (does not execute plans)
- Input: POPlan; Output: schedule
- Techniques (OR): PERT charts(precedence links, not causal), critical path method

Practical Planners Optimum-AIV (II)

STRIPS insufficient. Need to represent:

Complex situation: $\begin{cases} \surd \text{ hierarchies (to control complexity)} \\ \times \text{ quantification, conditional operators} \end{cases}$

Time: $\times$ durations of actions (not instantaneous)

Resources: $\times \begin{cases} \text{limitation, costs} \\ \text{sharable, reusable, etc.} \end{cases}$

OPTIMUM-AIV is based on **O-Plan** (Edinburgh), which can represent time, resources and hierarchical plans

Hierarchical decomposition

Action description

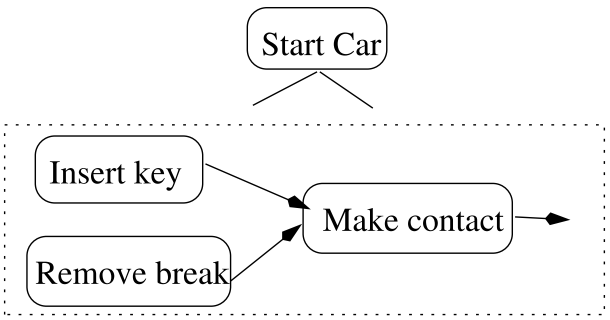
- High level: Go(SM), Buy(Milk), etc..
- Low level: Forward(1cm), Turn(1 deg), etc..

Which one to adopt?

- High level: are not executable
- Low level: combinatorial explosion

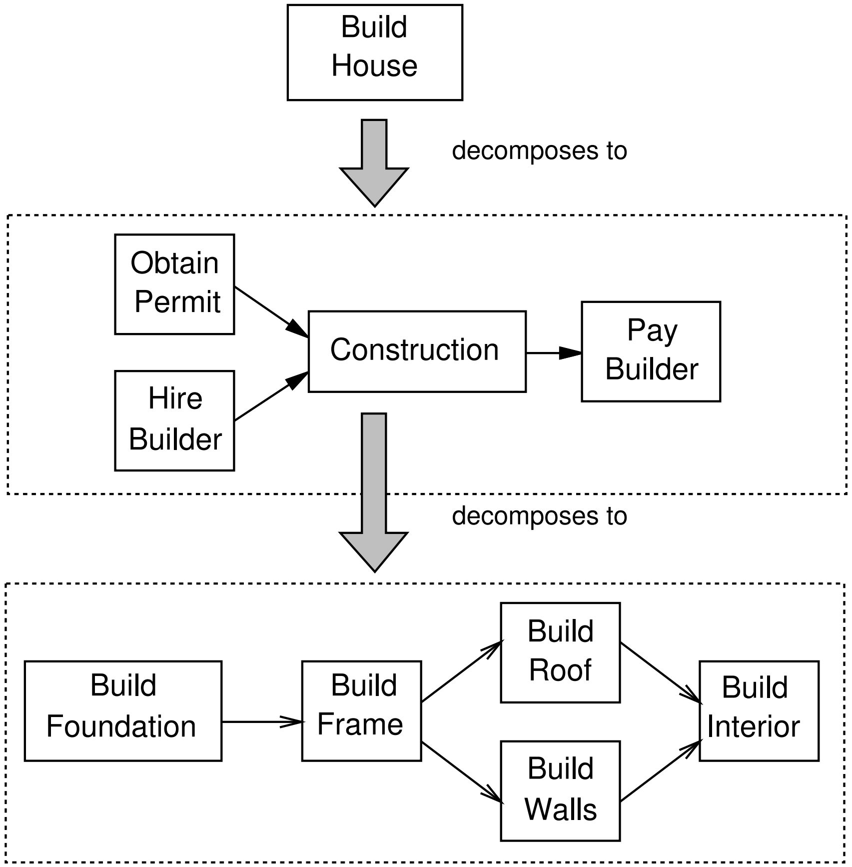
Hierarchical decomposition

An abstract operator is decomposed into a group of steps that forms a plan that implements the operator



Example: Build a house

An **abstract operator** can be decomposed iteratively..



⟶ Great opportunity: more than one decomposition per operator

Hierarchical decomposition

- Abstract operator
- **Primitive operator**: directly executable by the agent
- **Complete plan**: every step is a primitive operator

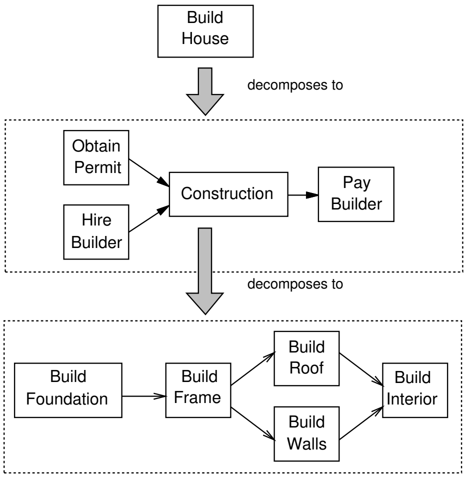
Two tasks:

1. Extend STRIPS to allow representation of abstract (nonprimitive) operators
2. Modify POPlanner to allow replacement of an nonprimitive operator by its decomposition

Extending the language (Task 1)

1. Partition operators into primitive/nonprimitive
   In general, distinction depends on the agent that will execute the plan
   e.g., contractor vs. worker
2. Provide a set of decomposition methods for each nonprimitive operator, more than one decomposition per operator
   $Decomose(o, p)$: nonprimitive operator o is decomposed into plan p

Nonprimitive operators (I)



Decompose (Construction,

Plan (Steps: $\{\ S_1 : Build(Foundation),$

$S_2 : Build(Frame), S_3 : Build(Roof),$

$S_4 : Build(Walls), S_3 : Build(Interior)\}$

Ordering: $\{S_1 \prec S_2 \prec S_3 \prec S_5, S_2 \prec S_4 \prec S_5\}$,

Bindings: $\{\}$,

Links: $\{S_1 \overset{Foundation}{\longrightarrow} S_2, S_2 \overset{Frame}{\longrightarrow} S_3, etc.\})$

Nonprimitive operators (II)

A plan p correctly implements an operator o if
p is a complete and consistent plan
for achieving the effects of o
given the preconditions of o

1. p is consistent (no contradictions in orderings or bindings)
2. Every effect of o asserted by at least one step of p (and not denied by some other, later step of p)
3. Every precondition of steps in p achieved by a step in p or by one of the preconditions of o

$\rightarrow$ guarantees proper hook-up after replacing o by p
$\rightarrow$ one still need to check for threats arising from interactions with other steps in the initial plan

$\longrightarrow$ complex plans can be built from simpler ones
$\longrightarrow$ plans can be generated, saved, re-used

Modifying the planner (Task 2)

POP $\longrightarrow$ HD-POP $\begin{cases} \text{1. decompose nonprimitive operators} \\ \text{2. input: plan (vs. goal)} \end{cases}$

```
function HD-POP(plan, operators, methods) returns plan
  inputs: plan, an abstract plan with start and goal steps (and possibly other steps)

  loop do
     if SOLUTION?(plan) then return plan
     S_need, c ← SELECT-SUB-GOAL(plan)
     CHOOSE-OPERATOR(plan, operators, S_need, c)
     S_nonprim ← SELECT-NONPRIMITIVE(plan)
     CHOOSE-DECOMPOSITION(plan, methods, S_nonprim)
     RESOLVE-THREATS(plan)
  end
```

`Solution?`: checks whether every step is primitive

`Select-nonprimitive`: arbitrarily selects a nonprimitive step

`Choose-decomposition`: selects a decomposition method

$\longrightarrow$ Fields of `Plan` updated accordingly

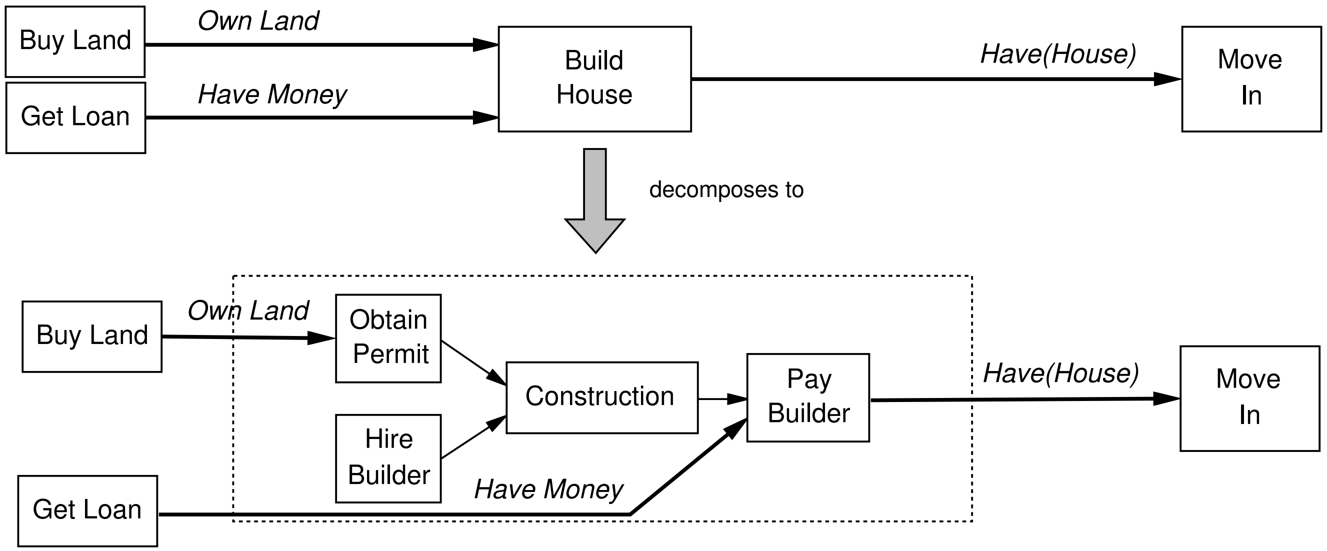
Updating the plan

Steps: remove nonprimitive step, add all steps of method

Bindings: Add all new bidings, fails if contradiction

Orderings: least commitment ($S_a \prec S_{nonprim}$: latest, $S_{nonprim} \prec S_z$: earliest), call `Resolve-Threats`

Links: ...



List of terms for glossary:

Due date: Monday, April 10, 2000

Words in margins in AIMA + ...

- Critical path method (outside AIMA)
- PERT charts, Gantt charts (outside AIMA)