Planning and Acting

by Stuart Russell

modified by Jacek Malec for LTH lecture
March 14, 2012

Chapter 11

Outline

- ♢ Planning and scheduling
- ♢ Hierarchical planning
- ♢ The real world
- ♢ Conditional planning
- ♢ Monitoring and replanning

Scheduling vs. planning

- ♢ Classical planning:
  what to do
  in what order
- ♢ but not:
  how long
  when
  using what resources
- ♢ Typical approach:
  plan first
  schedule later

Commonly used in real-world manufacturing and logistics

Representation

Job-shop scheduling problem

- ♢ a set of **jobs**
- ♢ each job is a collection of **actions** with some **ordering constraints**
- ♢ each action has a **duration** and a set of **resource constraints**
  resources may be **consummable** or **reusable**

- ♢ Solution:
  start times for all actions, obeying all constraints

Example problem description (Fig. 11.1)

Jobs((AddEngine1 $\prec$ AddWheels1 $\prec$ Inspect1),
(AddEngine2 $\prec$ AddWheels2 $\prec$ Inspect2))

Resources(EngineHoists(1), WheelStations(1), Inspectors(2), LugNuts(500))

Action(AddEngine1, DURATION: 30,
USE: EngineHoists(1))

Action(AddEngine2, DURATION: 60,
USE: EngineHoists(1))

Action(AddWheels1, DURATION: 30,
CONSUME: LugNuts(20), USE: WheelStations(1))

Action(AddWheels2, DURATION: 15,
CONSUME: LugNuts(20), USE: WheelStations(1))

Action(Inspect(i), DURATION: 10,
USE: Inspectors(1))

Solution

♢ may require optimisation of some complex utility/cost function

♢ simplest cases assume minimal-time criterion (makespan problem)

♢ method: identification of the **critical path** (CPM)
earliest times: forward sweep
latest times: backward sweep

♢ the pure temporal ordering may be solved in polynomial time (Fig. 11.2)

♢ a schedule involving resource constraints, i.e. disjunctive description, is NP-hard to find (Fig. 11.3)

♢ for complex scheduling problems it may be better to reconsider actions and constraints (thus interleaving planning with scheduling) — might lead to simpler scheduling problems

Hierarchical planning

♢ The key concept: **hierarchical decomposition**

♢ Hierarchical task networks (HTN), hierarchical planning

♢ High-level actions (HLA) have **refinements** (might be recursive)

♢ HTN planning:

```
Plan ← "Act"
repeat
    pick an HLA
    replace it with some refinement
    check whether the Plan achieves the goal
        if yes, return the Plan
```

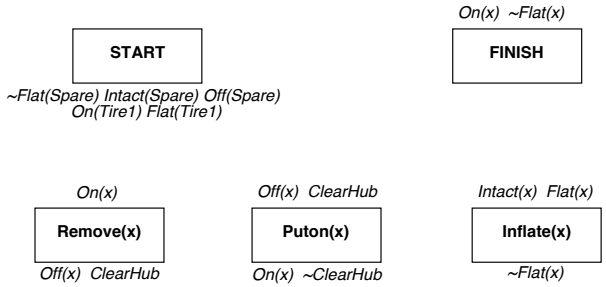

HTN planning: issues

♢ Finding good refinements
requires knowledge
usually domain-dependent knowledge is involved

♢ Learning successful refinements

♢ Approximating action descriptions
so that reachability needs not to be done
only on primitive action level

The real world



Things go wrong

Incomplete information

Unknown preconditions, e.g., $Intact(Spare)$?

Disjunctive effects, e.g., $Inflate(x)$ causes

$Inflated(x) \vee SlowHiss(x) \vee Burst(x) \vee BrokenPump \vee \ldots$

Incorrect information

Current state incorrect, e.g., spare NOT intact

Missing/incorrect postconditions in operators

Qualification problem:

can never finish listing all the required preconditions and possible conditional outcomes of actions

Solutions

Conformant or sensorless planning

Devise a plan that works regardless of state or outcome

Such plans may not exist

Conditional planning

Plan to obtain information (**observation actions**)

Subplan for each contingency, e.g.,

$[Check(Tire1),$

$\textbf{if}\ Intact(Tire1)\ \textbf{then}\ Inflate(Tire1)\ \textbf{else}\ CallMotormaennens]$

Expensive because it plans for many unlikely cases

Monitoring/Replanning

Assume normal states, outcomes

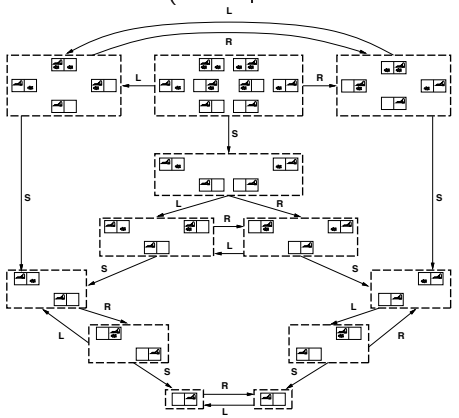
Check progress *during execution*, replan if necessary

Unanticipated outcomes may lead to failure (e.g., no M membership)

(Really need a combination; plan for likely/serious eventualities, deal with others when they arise, as they must eventually)

Conformant planning

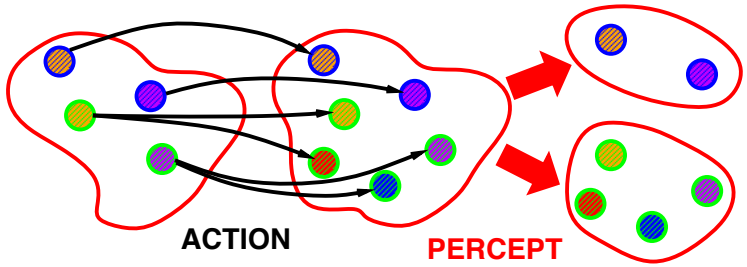
Search in space of belief states (sets of possible actual states)



Also called **sensorless planning**

Conditional planning

If the world is nondeterministic or partially observable
then percepts usually *provide information*,
i.e., *split up* the belief state



Conditional planning contd.

Conditional plans check (any consequence of KB +) percept

$[\ldots, \textbf{if } C \textbf{ then } Plan_A \textbf{ else } Plan_B, \ldots]$

Execution: check C against current KB, execute "then" or "else"

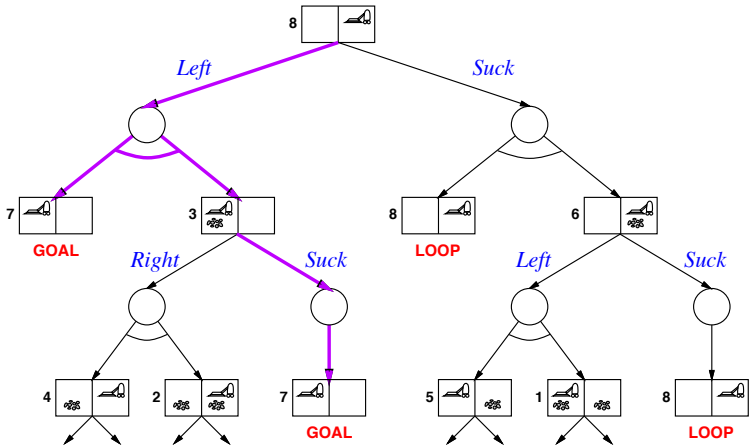
Need *some* plan for *every* possible percept

(Cf. game playing: *some* response for *every* opponent move)
(Cf. backward chaining: *some* rule such that *every* premise satisfied

AND–OR tree search (very similar to backward chaining algorithm)

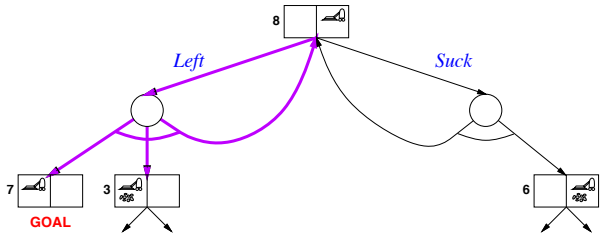
Example

Double Murphy: sucking or arriving may dirty a clean square



Example

Triple Murphy: also sometimes stays put instead of moving



$[L_1 : \ Left, \textbf{if } AtR \textbf{ then } L_1 \textbf{ else } [\textbf{if } CleanL \textbf{ then } [\,] \textbf{ else } Suck]]$

or $[\textbf{while } AtR \textbf{ do } [Left], \textbf{if } CleanL \textbf{ then } [\,] \textbf{ else } Suck]$

"Infinite loop" but will eventually work unless action always fails

Execution Monitoring

"Failure" = preconditions of *remaining plan* not met

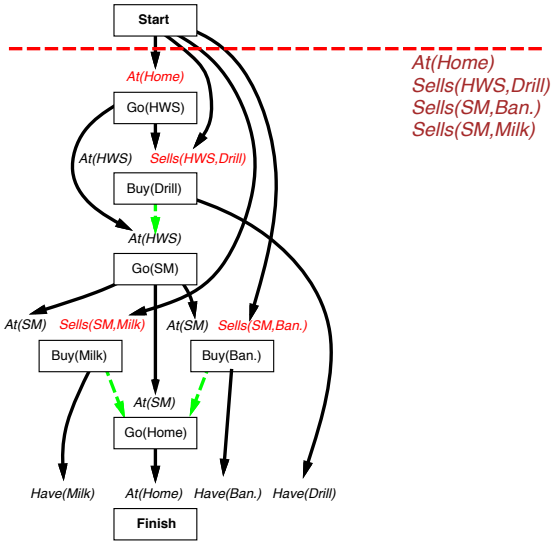
Preconditions of remaining plan
= all preconditions of remaining steps not achieved by remaining steps
= all causal links *crossing* current time point

On failure, resume POP to achieve open conditions from current state

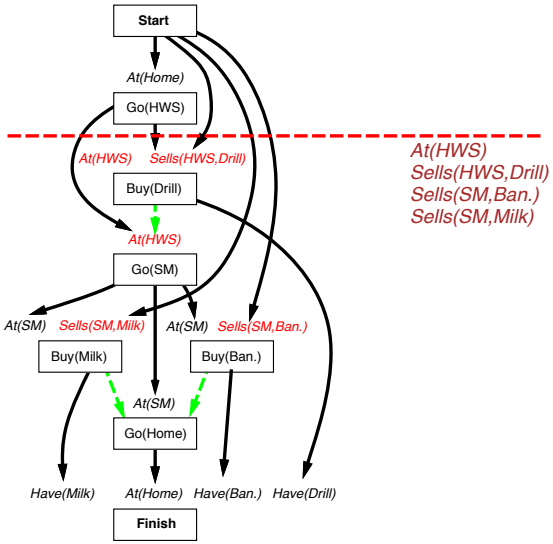
IPEM (Integrated Planning, Execution, and Monitoring):
keep updating $Start$ to match current state
links from actions replaced by links from $Start$ when done

Types of monitoring:
action monitoring (is it executable?)
plan monitoring (will the remaining plan succeed?)
goal monitoring (can I achieve it in some better way?)

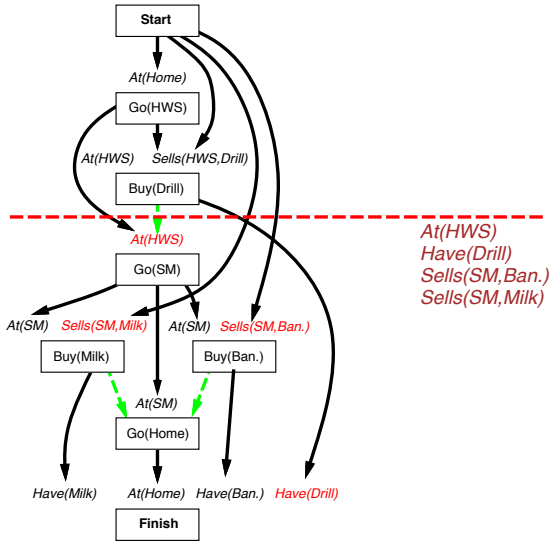
Example



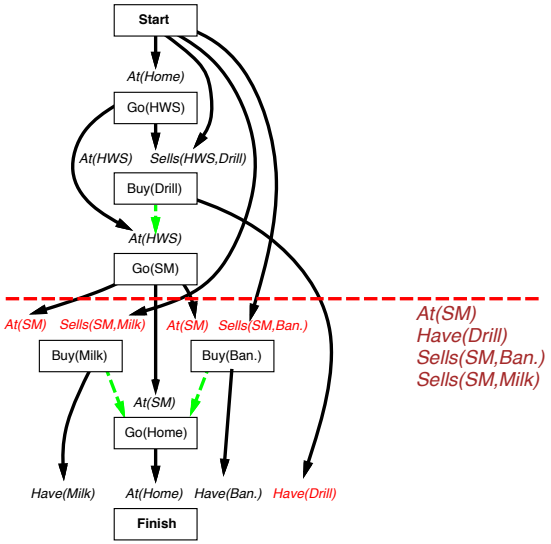
Example



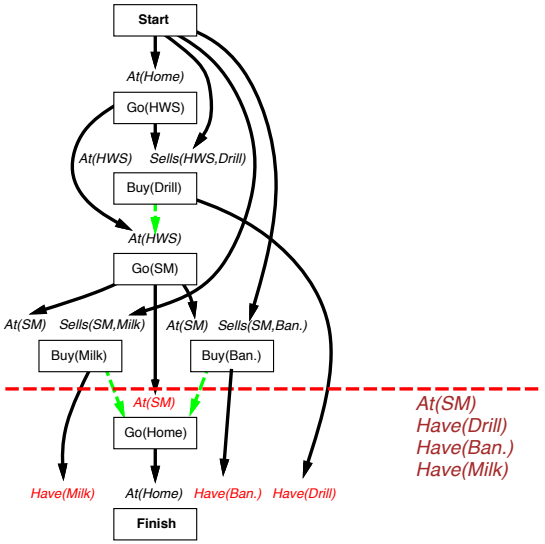
Example



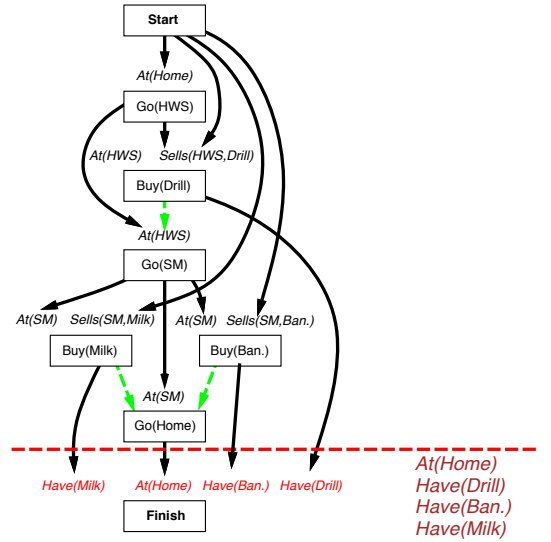
Example



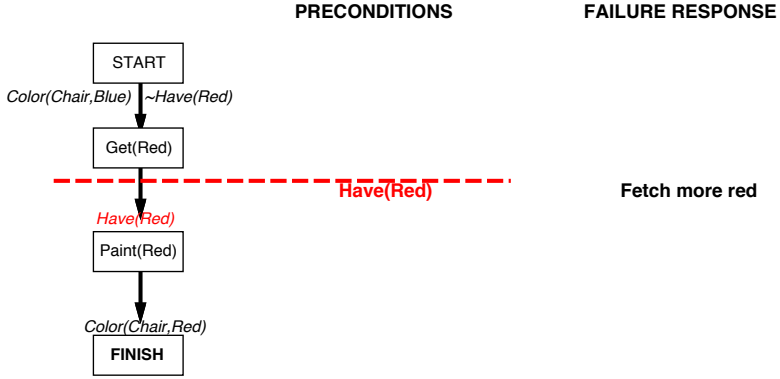
Example



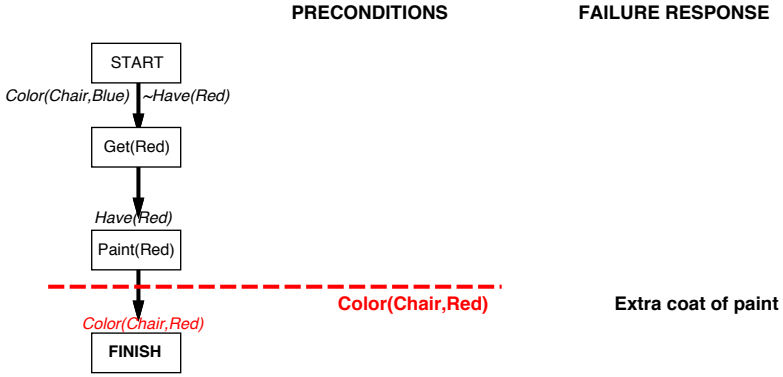
Example



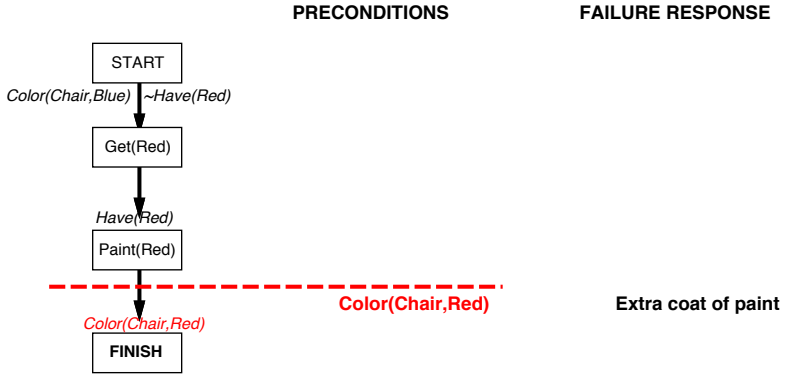
Emergent behavior



Emergent behavior



Emergent behavior



"Loop until success" behavior *emerges* from interaction between monitor/replan agent design and uncooperative environment

Multi-agent planning

- ♢ cooperative vs. competitive agents, communication
- ♢ resource sharing — coordination
- ♢ negotiation
- ♢ plan synchronisation

Assignment 2b

- ♢ Planning: PDDL 2.1
- ♢ test simple cases with existing descriptions
- ♢ apply PDDL to Wumpus world
- ♢ Have fun!