Planning, Execution, & Information Gathering

José Luis Ambite*

[* based on slides from Russell & Norvig, AIMA1]
The real world

START
~Flat(Spare) Intact(Spare) Off(Spare)
On(Tire1) Flat(Tire1)

On(x) ~Flat(x)

FINISH

On(x) Off(x) ClearHub
Remove(x) Puton(x)
Off(x) ClearHub On(x) ~ClearHub
Intact(x) Flat(x) Inflate(x)
~Flat(x)
Things go wrong

Incomplete information
   Unknown preconditions, e.g., \textit{Intact}(Spare)?
   Disjunctive effects, e.g., \textit{Inflate}(x) causes
   \textit{Inflated}(x) \lor \textit{SlowHiss}(x) \lor \textit{Burst}(x) \lor \textit{BrokenPump} \lor \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.,
   \[[\text{Check}(\text{Tire}_1), \text{if } \text{Intact}(\text{Tire}_1) \text{ then } \text{Inflate}(\text{Tire}_1) \text{ else } \text{Call AAA}]\]
   *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 AAA card)*

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

Start

At(Home)
Go(HWS)

At(HWS) Sells(HWS, Drill)
Buy(Drill)

At(HWS)
Go(SM)

At(SM) Sells(SM, Milk)
Buy(Milk)

At(SM) Sells(SM, Ban.)
Buy(Ban.)

At(SM)
Go(Home)

Have(Milk) At(Home) Have(Ban.) Have(Drill)

Finish

At(Home) Sells(HWS, Drill) Sells(SM, Ban.) Sells(SM, Milk)
Example

Start

At(Home)

Go(HWS)

At(HWS)  Sells(HWS, Drill)

Buy(Drill)

At(HWS)

Go(SM)

At(SM)  Sells(SM, Milk)

Buy(Milk)

At(SM)  Sells(SM, Ban.)

Buy(Ban.)

At(SM)

Go(Home)

Have(Milk)  At(Home)  Have(Ban.)  Have(Drill)

Finish

At(HWS)

Have(Drill)

Sells(SM, Ban.)

Sells(SM, Milk)
Example

Start
  ↓
  At(Home)
  ↓
  Go(HWS)
  ↓
  At(HWS)  Sells(HWS, Drill)
  ↓
  Buy(Drill)
  ↓
  At(HWS)
  ↓
  Go(SM)
  ↓
  At(SM)  Sells(SM, Milk)
  ↓
  Buy(Milk)
  ↓
  At(SM)  Sells(SM, Ban.)
  ↓
  Buy(Ban.)
  ↓
  At(SM)
  ↓
  Go(Home)
  ↓
  Have(Milk)  At(Home)  Have(Ban.)  Have(Drill)
  ↓
  Finish
  At(SM)
  Have(Drill)
  Have(Ban.)
  Have(Milk)
Planning & Execution

Initial State:

Move(x y)
pre: clear(x) ^ clear(y) ^ on(x z)
eff: on(x y) ^ clear(z) ^
¬ on(x z) ^ ¬ clear(y)

Goal:
On(C, D)
On(D, B)
Plan ready to start execution

but genie intervenes: moves D to B!

New state of the world:

Updated plan:
But it actually was a helpful interference:
• Can link to on(D B) from current state
• Move(D B) is now redundant
Now the agent can execute move(C, D) to achieve the goal. Unfortunately our agent is clumsy and drops C onto A instead of D. The new current state looks like:

And the updated plan is:
Keep planning to satisfy open condition on(C D)

Resulting plan:
Fortunately, this time execution works:

The plan is finally completed:
• Goals achieved
• No threats
• No unexecuted step “flaws”
Conditional Planning

Start

On(Tire1)
Flat(Tire1)
Inflated(Spare)

Finish
On(x)
Inflated(x)
(True)

Finish
On(Tire1)
Inflated(Tire1)
(True)

Start

On(Tire1)
Flat(Tire1)
Inflated(Spare)

Flat(Tire1)
Intact(Tire1)

Inflate(Tire1)
(Intact(Tire1))

Finish
On(Tire1)
Inflated(Tire1)

Start

On(Tire1)
Flat(Tire1)
Inflated(Spare)

Flat(Tire1)
Intact(Tire1)

Check(Tire1)

Intact(Tire1)
Intact(Tire1)

Inflate(Tire1)
(Intact(Tire1))

Finish
On(Tire1)
Inflated(Tire1)
(Intact(Tire1))
Planning and information gathering

- UWL representation language
- SENSEP algorithm