This lecture:

**Time:** Scheduling
Advanced Planning Concepts

This lecture:

**Time:** Scheduling

**Resources:** Consumables
This lecture:

**Time:** Scheduling

**Resources:** Consumables

**Active perception:** Look and feel?
This lecture:

- **Time:** Scheduling
- **Resources:** Consumables
- **Active perception:** Look and feel?
- **Hierarchical plans:** Abstracting
Exit survey: Game Theory

- Why don’t we take the mixed strategy if there is a dominant strategy?
- What advantage is gained by a player by looking irrational?

Entry survey: Advanced Planning (0.25 points of final grade)

- Do you think a classical planning planning approach can be used for solving scheduling problems?
- What would be the advantage of making hierarchical plans?
I’m Late Again!

- Fixed times for day start and class
- Durations
  - Wake up: 10 minutes
  - Eat: 30 minutes

Earliest and latest start times of each event?

- Earliest (Wake up) = 8am
- Latest (Wake up) = ?
- Earliest (Eat) = ?
- Latest (Eat) = 10:00 - 00:30 = 9:30 am
I’m Late Again!

- Fixed times for day start and class
- Durations
  - Wake up: 10 minutes
  - Eat: 30 minutes

Earliest and latest start times of each event?

- Earliest (Wake up) = 8 am
- Latest (Wake up) = ?
- Earliest (Eat) = ?
- Latest (Eat) = 10:00 - 00:30 = 9:30 am

Earliest and latest start times of each event?
I’m Late Again!

- Fixed times for day start and class
- Durations
  - Wake up: 10 minutes
  - Eat: 30 minutes

Earliest and latest start times of each event?
- Earliest(Wake up) = 8am
- Latest(Wake up) = ?
- Earliest(Eat) = ?
- Latest(Eat) = 10:00 - 00:30 = 9:30am
Multiple Paths to Finish: Car with 2 Engines

- Critical path?
Multiple Paths to Finish: Car with 2 Engines

Earliest(Start) = 0
Earliest(B) = max A → B Earliest(A) + Duration(A)
Latest(A) = max A ← B Latest(B) − Duration(A)
Latest(Finish) = Earliest(Finish)
Multiple Paths to Finish: Car with 2 Engines

Earliest(Start)=0
Earliest(B)=\max_{A\rightarrow B} \text{Earliest}(A) + \text{Duration}(A)
Latest(A)=\max_{A\leftarrow B} \text{Latest}(B) - \text{Duration}(A)
Latest(Finish)=\text{Earliest}(Finish)
Action(Inspect(n₁, n₂, n₃, n₄, n₅, b₁, b₂, b₃, b₄, b₅)),
  PRE: Fastened(n₁, b₁), ..., Fastened(n₅, b₅)),
  EFF: Inspected)
Action(Fasten(n, b),
  PRE: Nut(n) ∧ Bolt(b)
  EFF: Fastened(n, b) ∧ ¬Nut(n) ∧ ¬Bolt(b))
Init(Nut(N1), ..., Nut(N₄),
     Bolt(B₁), ...Bolt(B₅))
Goal(Inspected)
Resources: Can We Use Action Schemas?

Action(Inspect(n_1, n_2, n_3, n_4, n_5, b_1, b_2, b_3, b_4, b_5)),
PRE: Fastened(n_1, b_1), ..., Fastened(n_5, b_5)),
EFF: Inspected)
Action(Fasten(n, b),
PRE: Nut(n) \land Bolt(b)
EFF: Fastened(n, b) \land \neg Nut(n) \land \neg Bolt(b))
Init(Nut(N1), ..., Nut(N4),
    Bolt(B1), ...Bolt(B5))
Goal(Inspected)

- Will it reach the goal?
Resources: Can We Use Action Schemas?

- Will it reach the goal? **No**, one nut is missing.
Resources: Can We Use Action Schemas?

- Will it reach the goal? **No**, one nut is missing.
- Depth first tree search: how many paths to eval?
  - **Small**: 1, 4, 5 ?
  - **Medium**: 4 + 5 or 4 × 5 ?
  - **Large**: 4!, 5!, or 4! × 5! ?

Action(Inspect(n₁,n₂,n₃,n₄,n₅,b₁,b₂,b₃,b₄,b₅)),
PRE: Fastened(n₁, b₁), ..., Fastened(n₅, b₅)),
EFF: Inspected )
Action(Fasten(n, b),
PRE: Nut(n) ∧ Bolt(b)
EFF: Fastened(n, b) ∧ ¬Nut(n) ∧ ¬Bolt(b))
Init(Nut(N1), ..., Nut(N4),
    Bolt(B1), ...Bolt(B5))
Goal(Inspected)
Resources: Can We Use Action Schemas?

Will it reach the goal? No, one nut is missing.

Depth first tree search: how many paths to eval?

- Small: 1, 4, 5?
- Medium: 4 + 5 or 4 × 5?
- Large: 4!, 5!, or 4! × 5!.

Really inefficient!
Modify Action Schemas to Optimize Resource Problems

No need to try combinations of same resources.

Define:

- **Resources**: Specify quantity.
- **Use**: Specify requirement.
- **Consume**: Removes resource.

Günay

Ch. 11, Advanced Planning
No need to try combinations of same resources.

Define:

- **Resources**: Specify quantity.
- **Use**: Specify requirement.
- **Consume**: Removes resource.

```
Action(Inspect(n_1, n_2, n_3, n_4, n_5, b_1, b_2, b_3, b_4, b_5),
PRE: Fastened(n_1, b_1), ..., Fastened(n_5, b_5)),
USE: Inspector(1)
EFF: Inspected)

Action(Fasten(n, b),
CONSUME: Nuts(1), Bolts(1)
EFF: Fastened(n, b)
Resources(Nuts(5), Bolts(4), Inspectors(1))
```
No need to try combinations of same resources.

Define:

**Resources**: Specify quantity.
**Use**: Specify requirement.
**Consume**: Removes resource.

No exponential explotion anymore!

\begin{verbatim}
Action(Inspect(n_1, n_2, n_3, n_4, n_5, b_1, b_2, b_3, b_4, b_5),
PRE: Fastened(n_1, b_1), ..., Fastened(n_5, b_5)),
USE: Inspector(1)
EFF: Inspected )
Action(Fasten(n, b),
CONSUME: Nuts(1), Bolts(1)
EFF: Fastened(n, b)
Resources(Nuts(5), Bolts(4), Inspectors(1))
\end{verbatim}
Hierarchical Planning: Abstraction

Remember Stanley:

- **High-level goal:**
  - Reach target at GPS coordinates
  - Drive on road

- **Low-level actions:**
  - Adjust steering wheel
  - Press/release gas/break pedals
Hierarchical Planning: Abstraction

Remember Stanley:

- **High-level goal:**
  - Reach target at GPS coordinates
  - Drive on road

- **Low-level actions:**
  - Adjust steering wheel
  - Press/release gas/break pedals

How to connect

1. high-level (abstract) planning with
2. low-level planning?
Hierarchical Planning: Abstraction

Remember Stanley:

- **High-level goal:**
  - Reach target at GPS coordinates
  - Drive on road

- **Low-level actions:**
  - Adjust steering wheel
  - Press/release gas/break pedals

How to connect

1. high-level (abstract) planning with
2. low-level planning?

Solution: Refinement
Refining Abstractions

Multiple ways to refine abstractions:

Refinement (Go(Home, SFO),
STEPs: [Drive(Home, SFOLongTermParking),
Shuttle(SFOLongTermParking, SFO)])

Refinement (Go(Home, SFO),
STEPs: [Taxi(Home, SFO)])

Refinement (Navigate([a, b], [x, y])
PRECOND: a = x \land b = y
STEPs: [])

Refinement (Navigate([a, b], [x, y])
PRECOND: Connected([a, b], [a - 1, b])
STEPs: [Left, Navigate([a - 1, b], [x, y])])

Refinement (Navigate([a, b], [x, y])
PRECOND: Connected([a, b], [a + 1, b])
STEPs: [Right, Navigate([a + 1, b], [x, y])])
Hierarchical Planning: Reachable States

Reachable?

No.

Found solution: Now backtrack from solution.
Hierarchical Planning: Reachable States

Reachable?

No.
Hierarchical Planning: Reachable States

Reachable?

No.

Found solution:
Hierarchical Planning: Reachable States

Reachable?

No.

Found solution:

Now backtrack from solution.
Reachable States Question

Estimates of refinement:

**Underestimate:** We reach it for sure.

**Overestimate:** Possibly reachable.

Below examples:

- **Reachable? Yes, No, Maybe?**
Reachable States Question

Estimates of refinement:

**Underestimate:** We reach it for sure.

**Overestimate:** Possibly reachable.

Below examples:

- Reachable? Yes, No, Maybe?

No

Yes.
Reachable States Question

Estimates of refinement:

**Underestimate:** We reach it for sure.

**Overestimate:** Possibly reachable.

Below examples:

- **Reachable?** Yes, No, Maybe?

  ![Diagram](image_url)

  **No**

  **Yes.**
Reachable States Question

Estimates of refinement:

**Underestimate:** We reach it for sure.

**Overestimate:** Possibly reachable.

Below examples:

- Reachable? Yes, No, Maybe?

![Diagram](image-url)

No  Yes.  Maybe.
Sometimes the agent needs to look first.
Sometimes the agent needs to look first.

Init(Object(Table) \land Object(Chair) \land Can(C_1) \land Can(C_2) \land InView(Table))
Goal(Color(Chair, c) \land Color(Table, c))

Action(\text{RemoveLid}(\text{can}),)
\hspace{1cm}\text{PRECOND: Can(\text{can})}
\hspace{1cm}\text{EFFECT: Open(\text{can})})

Action(\text{Paint}(x, \text{can}),)
\hspace{1cm}\text{PRECOND: Object(x) \land Can(\text{can}) \land Color(\text{can, c}) \land Open(\text{can})}
\hspace{1cm}\text{EFFECT: Color(x, c))}

Percept(Color(x, c),)
\hspace{1cm}\text{PRECOND: Object(x) \land InView(x)}

Percept(Color(\text{can, c}),)
\hspace{1cm}\text{PRECOND: Can(\text{can}) \land InView(\text{can}) \land Open(\text{can})}

Action(\text{LookAt}(x),)
\hspace{1cm}\text{PRECOND: InView(y) \land (x \neq y)}
\hspace{1cm}\text{EFFECT: InView(x) \land \neg InView(y))}