Real World Planning and Acting

CS4246/CS5446

AI Planning and Decision Making

Hierarchical Planning and Acting

Manage complexity

Examples

- Example 1:

  How to go to CSxx46 lecture from home?

  - Solution: Go to i3 from home, find lecture hall
  - Solution: take MRT, change to internal bus, get off, find i3-AUD.

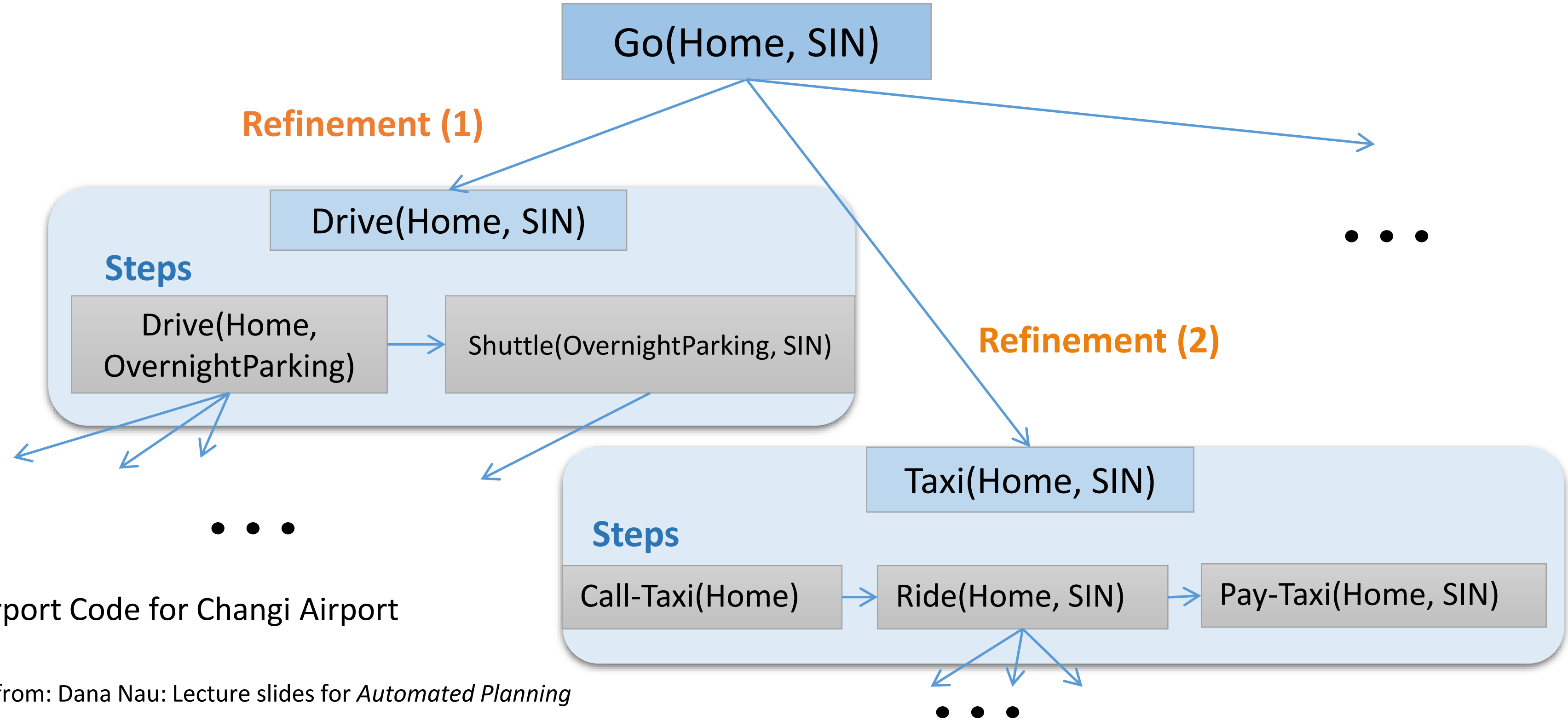
- Example 2:

  How to land on the Jezero Crater of Mars from X space station?

- Example 3:

  How to get from home to Changi airport?

Example: Going to Changi Airport



SIN: Airport Code for Changi Airport

Adapted from: Dana Nau: Lecture slides for *Automated Planning*

Managing Complexity in Planning

- Hierarchical decomposition
  - Division of tasks into different subtasks at next level
  - At each level focus only a small number of tasks
- Deferred planning
  - Planning can occur before and during plan execution
  - Particular action can remain at an abstract level prior to the execution phase

Hierarchical Decomposition

- Key benefits
  - At each level of hierarchy, a task is reduced to a small number of subtasks or activities at the next lower level
  - Computational cost of finding the correct way to arrange activities for current problem is small
- Examples
  - Software components, subroutines
  - Military, government, and corporations

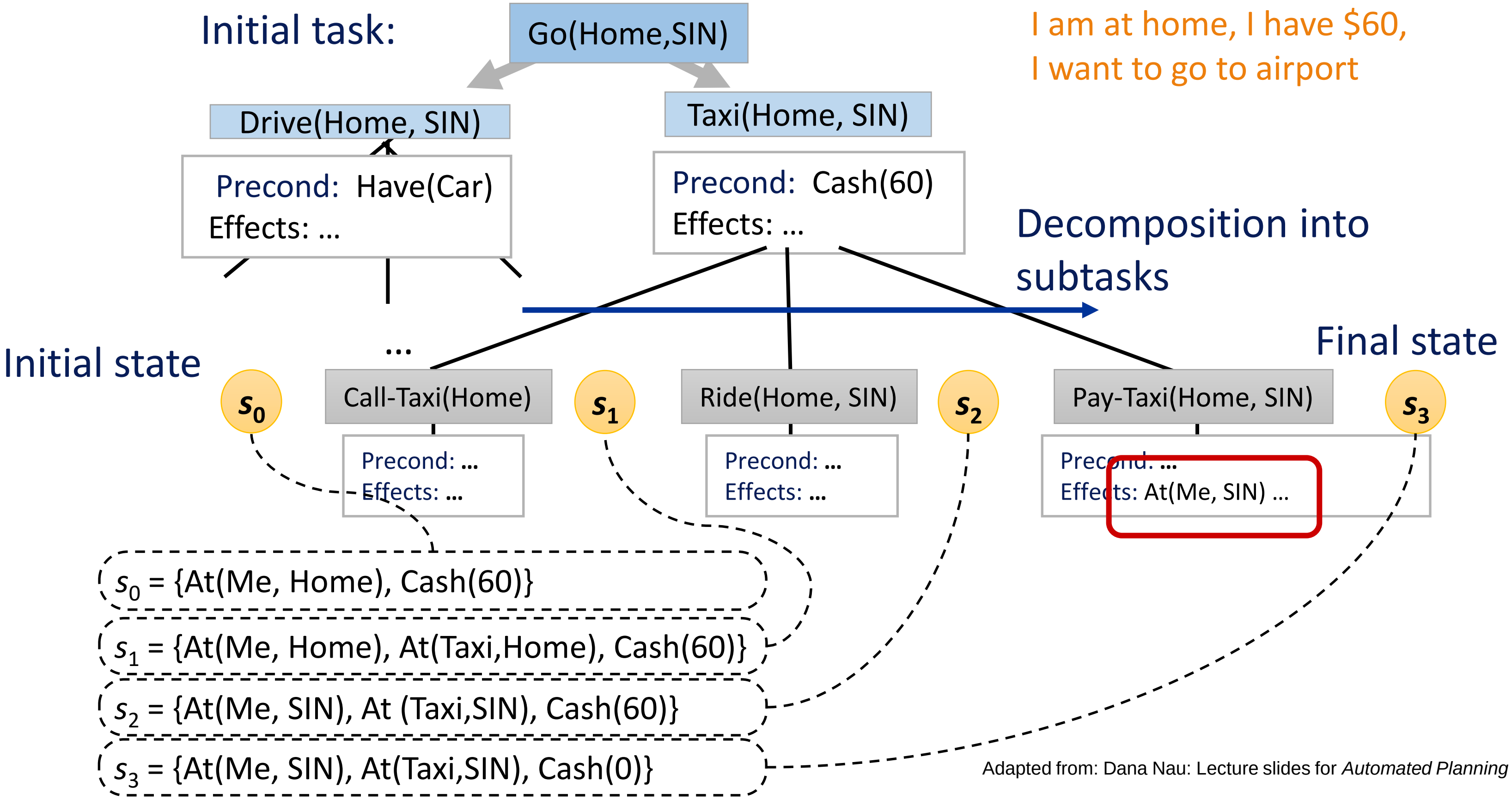
Hierarchical Task Networks

- Hierarchical task networks (HTNs)
  - A formalism to help understand hierarchical decomposition
  - A planning model that manages complexity through task abstractions
- Key concept
  - High-level actions (HLAs)
- Assumptions
  - Full observability
  - Deterministic
  - Availability of primitive actions with standard precondition-effect schemas
  - Main ideas are general in problem solving and planning and decision making

HTN Planning

- Planning Problem or Model
  - HLAs, action schemas, initial state, goal state, task list
- Planning Algorithm
  - Input: a problem
  - Output: a solution in the form of an action sequence
- Planning Solution
  - Any executable plan generated by recursively applying:
    - HLA to nonprimitive tasks
    - Actions to primitive tasks
  - A goal state that satisfies certain properties

Example: Going to Changi Airport



Adapted from: Dana Nau: Lecture slides for *Automated Planning*

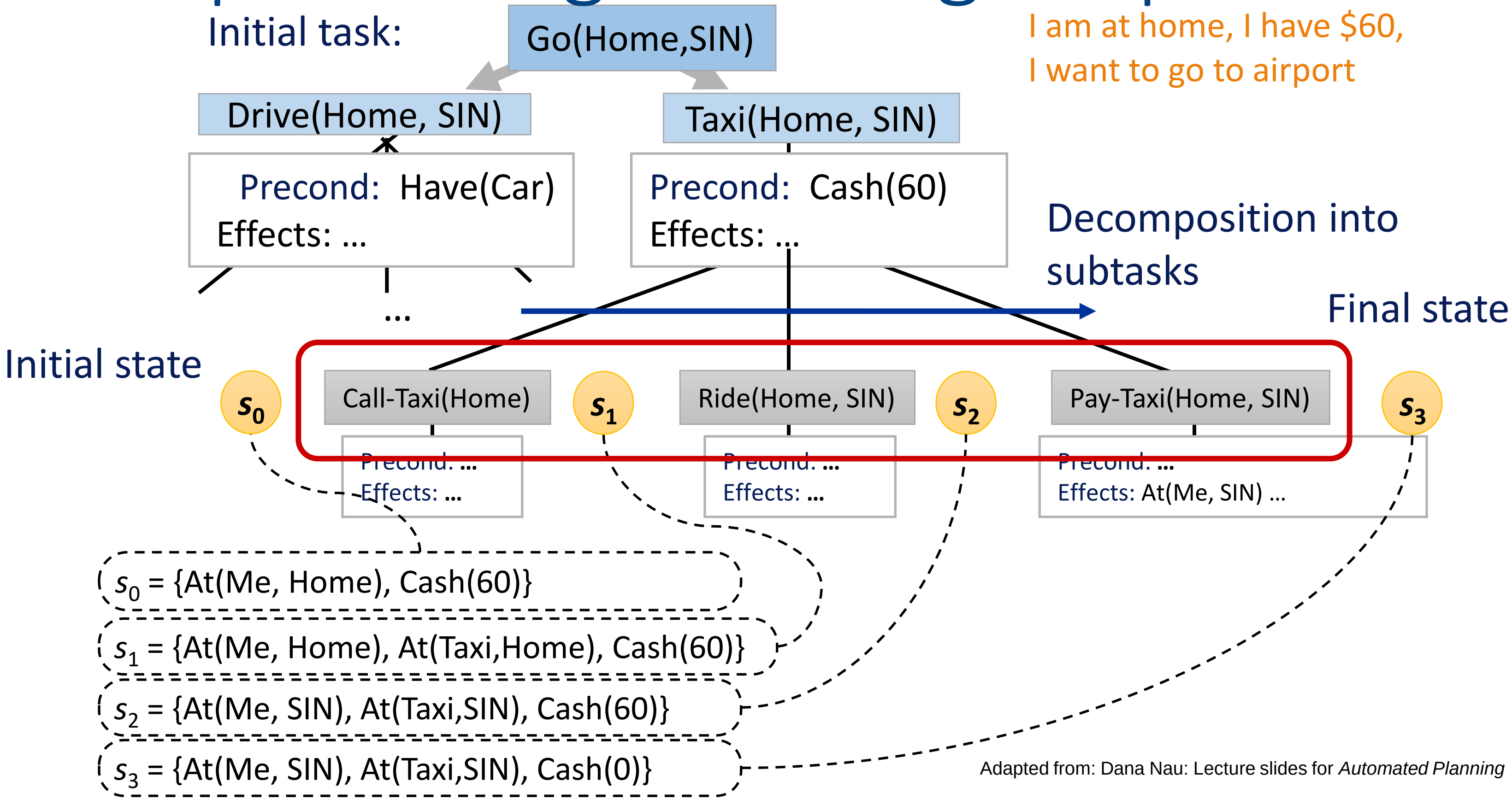
High-Level Actions (HLAs)

- Definition
  - Each HLA has one or more refinements into a sequence of actions
  - Each (refined) action can be an HLA or a primitive action
  - Recursive refinement may be needed
- Meaning
  - HLAs and their refinements embody knowledge about how to do things
    e.g., Go(Home, SIN) – drive or take a taxi

Implementation

- HLA implementation
  - An HLA refinement that contains only primitive actions
- High-level plan implementation
  - High-level plan – a sequence of HLAs
  - Concatenate implementations of each HLA in the sequence
- Observation
  - Given the precondition-effect definitions of each primitive action, can directly determine whether any given implementation of a high-level plan achieves the goal.

Example: Going to Changi Airport



Adapted from: Dana Nau: Lecture slides for *Automated Planning*

Planning with HLAs

- Definition
  - Achieves the goal from a given state if at least one of its implementations achieves the goal from that state
- Note
  - Not all implementations need to achieve the goal
  - The agent decides which implementation to execute
- Question:
  - How is this different from nondeterministic planning?

Planning with HLAs

- With one HLA implementation
  - Compute preconditions and effects of HLA from those of the implementation
  - Treat HLA exactly as if it were a primitive action
- Observation
  - Right collection of HLAs can reduce time complexity of (blind) search from exponential to linear in solution depth
  - Devising an appropriate collection of HLAs is HARD!
- With multiple HLA implementations
  - Search among implementations for one that works; OR
  - Reason directly about the HLAs - enables derivation of provably correct abstract plans, without having to consider their implementations

Real World Planning and Acting

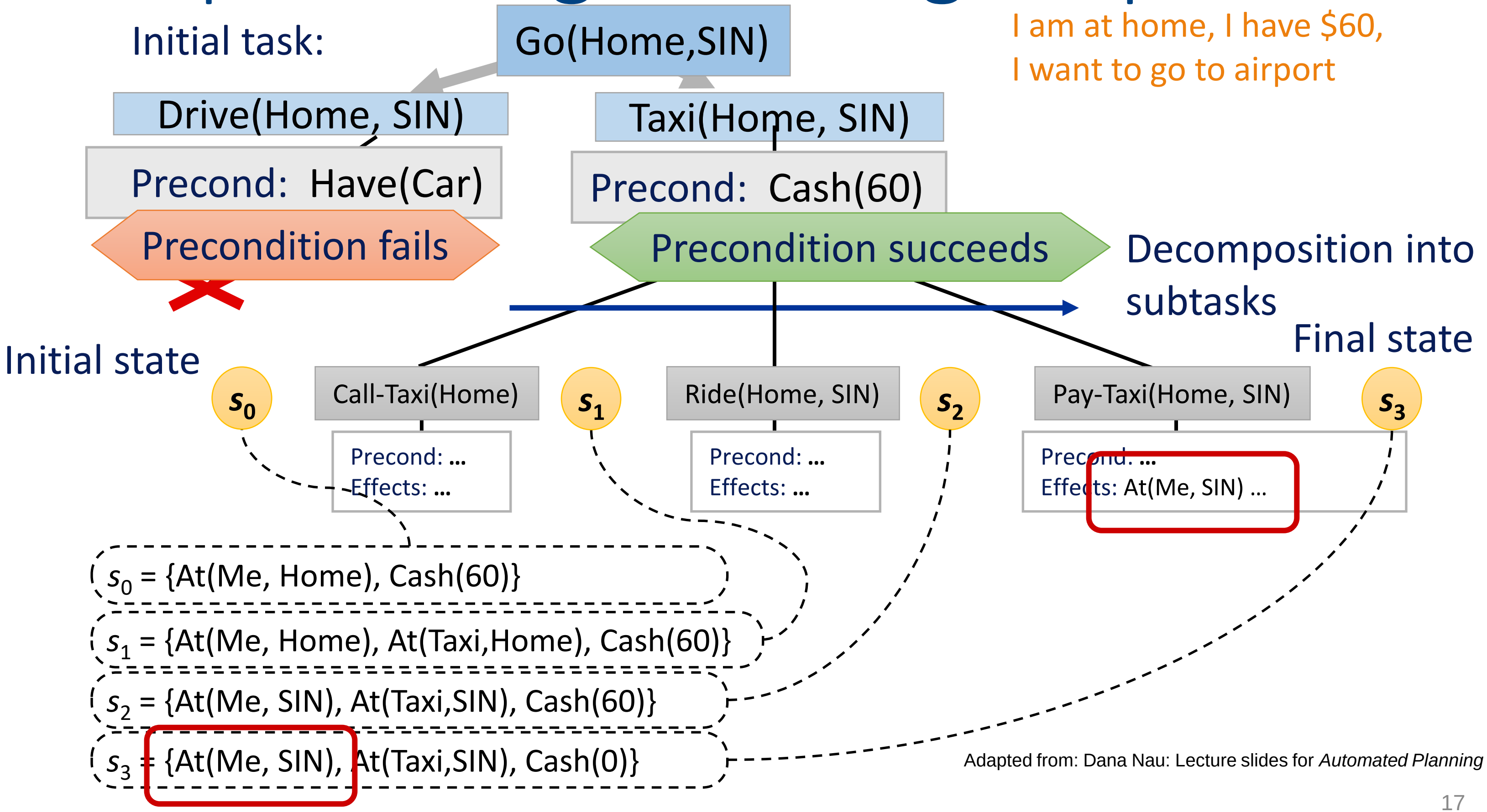
CS4246/CS5446

AI Planning and Decision Making

Hierarchical Planning as Search

Searching for Primitive Solutions

Example: Going to Changi Airport



Adapted from: Dana Nau: Lecture slides for *Automated Planning*

Searching for Primitive Solutions

- HTN Planning
  - Start with top level action Act
  - Find an implementation of Act that achieves the goal
- Hierarchical planning algorithm
  - Repeatedly choose an HLA in current plan and replace with refinement
  - Until the plan achieves the goal
- Example:
  - Breadth-first search tree
  - Plans are considered in order of depth of nesting of the refinements, rather than number of primitive steps
  - Can use graph-search, depth-first, and iterative deepening

Generic Planning Framework

- Classical planning definition:
  - For each primitive action a_i:
  - Provide one refinement of Act with steps $-\ [ai, Act]$
  - Create recursive definition of Act to add actions
  - Final refinement:
    - steps – empty, precondition – goal, effect – null
- Algorithm:
  - Repeatedly choose an HLA in the current plan
  - Replace it with one of its refinements
  - Until the plan achieves the goal

Hierarchical Search

```
function HIERARCHICAL-SEARCH(problem, hierarchy) returns a solution or failure
  frontier ← a FIFO queue with [Act] as the only element
  while true do
      if IS-EMPTY(frontier) then return failure
      plan ← POP(frontier)          // chooses the shallowest plan in frontier
      hla ← the first HLA in plan, or null if none
      prefix,suffix ← the action subsequences before and after hla in plan
      outcome ← RESULT(problem.INITIAL, prefix)
      if hla is null then          // so plan is primitive and outcome is its result
          if problem.IS-GOAL(outcome)  then return plan
      else for each sequence in REFINEMENTS(hla, outcome, hierarchy) do
          add APPEND(prefix, sequence, suffix) to frontier
```

Source: RN Figure 11.8

Hierarchical Search

- Main idea:
  - Explore space of sequences that conform to knowledge in the HLA library about how things are to be done
  - Knowledge about the problem is encoded in action sequences in each refinement and in the preconditions of the refinements
- Practical impact:
  - Can generate huge plans with little search
    - e.g., O-PLAN to develop production plans for HITACHI (Bell and Tate 1995)
  - Hierarchically structured – easier for human to understand
  - Find out more about recent applications in use!

Complexity Analysis

- Assumption
  - A planning problem has a solution with d primitive actions.
- For non-hierarchical, forward state-space planner
  - With b allowable actions at each state, cost is $O(b^d)$
- For HTN planner
  - Suppose each nonprimitive action has r possible refinements, each into k actions at the next lower level
  - So $r^{(d-1)(k-1)}$ possible regular decomposition trees could be constructed (see details in RN 11.4.2)
- Observation
  - Small r and large k - library of HLAs with small number of refinements each yielding a long action sequence - May be hard to construct!

NUS National University of Singapore | Computing

Real World Planning and Acting

CS4246/CS5446

AI Planning and Decision Making

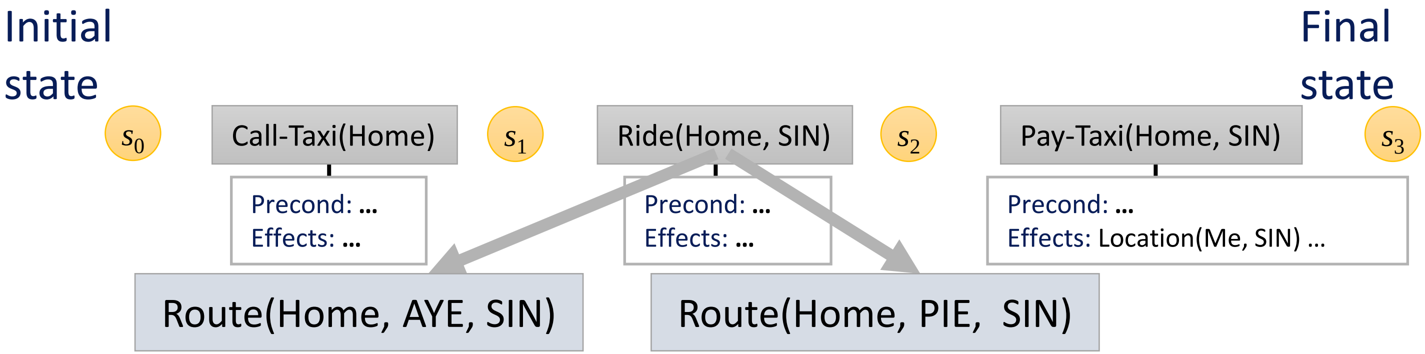
Proving Plan Properties

Searching for Abstract Solutions

Motivation

- Example:
  - We should determine if a high-level plan can get one to the airport, without going through all the specific details like precise route or alighting terminal

    [Call-Taxi(Home), Ride(Home, SIN), Pay-Taxi(SIN)]



Searching for Abstract Solutions

- Approach
  - Write precondition-effect description of the HLAs
  - Prove that the high-level plan achieves the goal
  - Work in small search space of high-level actions
  - Refine committed plan to achieve exponential reduction

Searching for Abstract Solutions

- Downward refinement property (of HLA descriptions)
  - Through description of the steps:
  - Every high-level plan that “claims” to achieve the goal achieves the goal
  - At least one implementation achieves the goal
- Main challenges
  - How to write HLAs with downward refinement property?
  - How to write HLAs with multiple implementations?
  - How to describe effects of an action that can be implemented in many different ways?
- Key idea
  - Determine if reachable sets of a sequence of HLAs in the plan overlap with goals

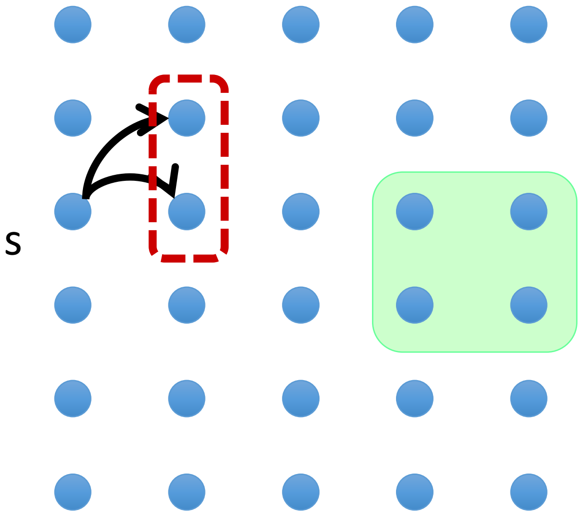
Reachable Set

- Reachable set of an HLA
  - Given a state s and an HLA h: $REACH(s, h)$ is the set of states reachable by any of the HLA's implementations

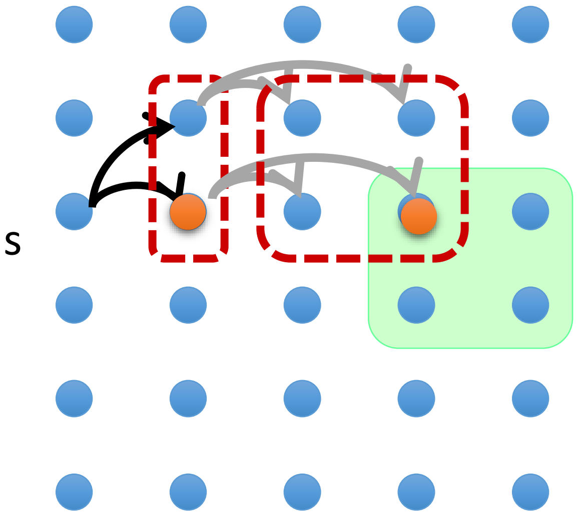
- Reachable set of a sequence of HLAs
  - Reachable set of a sequence of HLAs $[h_1, h_2]$ is the union of all the reachable sets obtained by applying h_2 in each state in the reachable set of h_1:

$$REACH(s, [h_1, h_2]) = \cup_{s' \in REACH(s, h_1)} REACH\ (s', h_2)$$

Example



$REACH(s, [h_1])$



$REACH(s, [h_1, h_2])$

Source: RN: Fig 11.9

High-Level Planning

- Practical implications
  - Agent can choose element of the reachable set it ends up in when it executes the HLA
  - HLA with multiple refinements is more “powerful” than the same with fewer refinements
- High-level plan
  - A sequence of HLAs
  - Achieves goal if its reachable set intersects set of goal states
  - Otherwise, the plan does not work
- Search algorithm
  - Search among high-level plans
  - Look for one whose reachable set intersects goal
  - Once that happens, commit to that abstract plan
  - Focus on refining the plan further

Representing HLA Effects

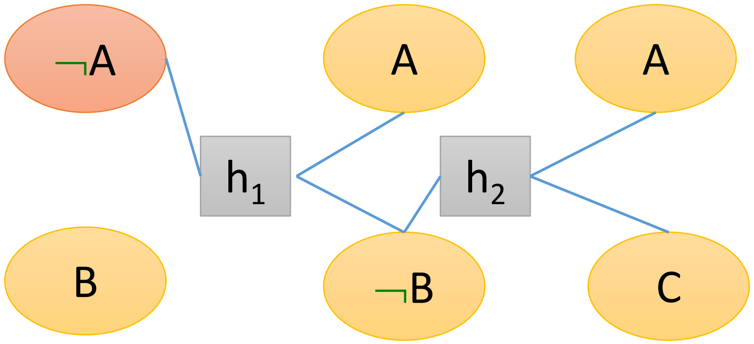
- Effects as reachable sets
  - As reachable set for each possible initial state
  - Represent changes made to each fluent or state variable
- Recall: Primitive action
  - Can add or delete a fluent or variable or leave it unchanged
- HLA
  - Can also control variable value, depending on implementation chosen
  - Description derivable, in principle, from descriptions of its refinements, such that the downward refinement property holds

Representing Reachable Set

- Notations:
  - $\sim$ means possibly, if the agent chooses
  - E.g., $\tilde{+}A$ means "possibly add A", i.e., either leave A unchanged or make it True
- Questions:
  - What do $\tilde{-}A$ and $\tilde{\pm}A$ mean?
- Example

  $Go(Home, SIN)$ with two refinements
  - $Drive(Home, SIN)$ and $Taxi(Home, SIN)$
  - Possibly delete $Cash$ (if agent decides to take a taxi)
  - So should have effect $\tilde{-}Cash$

Example



- Consider:
  - Schemas for HLAs h_1 and h_2:
  - Action (h_1, Precond: $\neg A$, Effect: $A \wedge \widetilde{-} B$)
  - Action (h_2, Precond: $\neg B$, Effect: $\widetilde{+} A \wedge \widetilde{\pm} C$)
- Meaning:
  - h_1 adds A and possibly delete B
  - h_2 possibly adds A and has full control over C
- Exercise:
  - If only B is true in the initial state and goal is $A \wedge C$
  - Q: What sequence of HLAs will achieve the goal?

HTN Planning Today

- Key idea
  - Construct plan library (knowledge base) of known methods for implementing complex, HLAs
- Approach
  - Learn planning methods from problem-solving experience
  - Save used plan in library as a method for task-specific HLA implementation
  - Accumulate knowledge over time
  - Generalize methods, eliminating problem-specific details, keeping key elements of the plan
- In practice:
  - Many real-world applications; ideas adopted in modern day planning and reinforcement learning
  - Old HTN planners: Noah, Nonlin, O-Plan, SIPE, SIPE-2, SHOP, SHOP2
  - Fast Downward (Helmert 2006) won 2004 IPC; uses hierarchical decomposition of planning tasks to derive heuristics with delayed evaluation in best first search
  - New research trends in hierarchical planning and hierarchical reinforcement learning

Example: PANDA

- The PANDA framework for hierarchical planning
  - https://rdcu.be/cn6Ra
  - Höller, D., et al., *The PANDA Framework for Hierarchical Planning*. KI - Künstliche Intelligenz, 2021.
  - Höller, D., et al., *HDDL: An Extension to PDDL for Expressing Hierarchical Planning Problems*. Proceedings of the AAAI Conference on Artificial Intelligence, 2020. **34**(06): p. 9883-9891.

Summary

- Hierarchical planning
  - Using abstraction to manage complexity
  - Planning as refinements
  - Planning in abstract space
- HTN Planning
  - Focus on tasks instead of goals
  - Use hierarchical decomposition and delayed planning ideas to manage complexity
- Searching for primitive actions
  - Recursive refinement
- Search for abstract actions
  - Downward refinement property
  - Check if reachable set intersects with goals