CAP6671 Intelligent Systems

Lecture 4:

Planning in Computer Games

Instructor: Dr. Gita Sukthankar
Email: gitars@eecs.ucf.edu

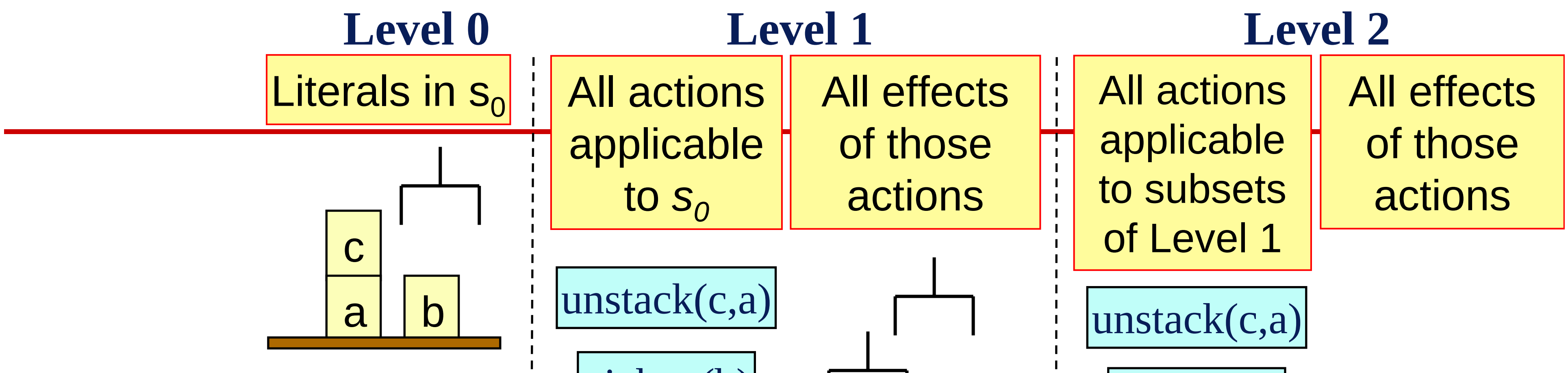
Schedule: T & Th 9:00-10:15am

Location: HEC 302

Office Hours (in HEC 232):
T & Th 10:30am-12

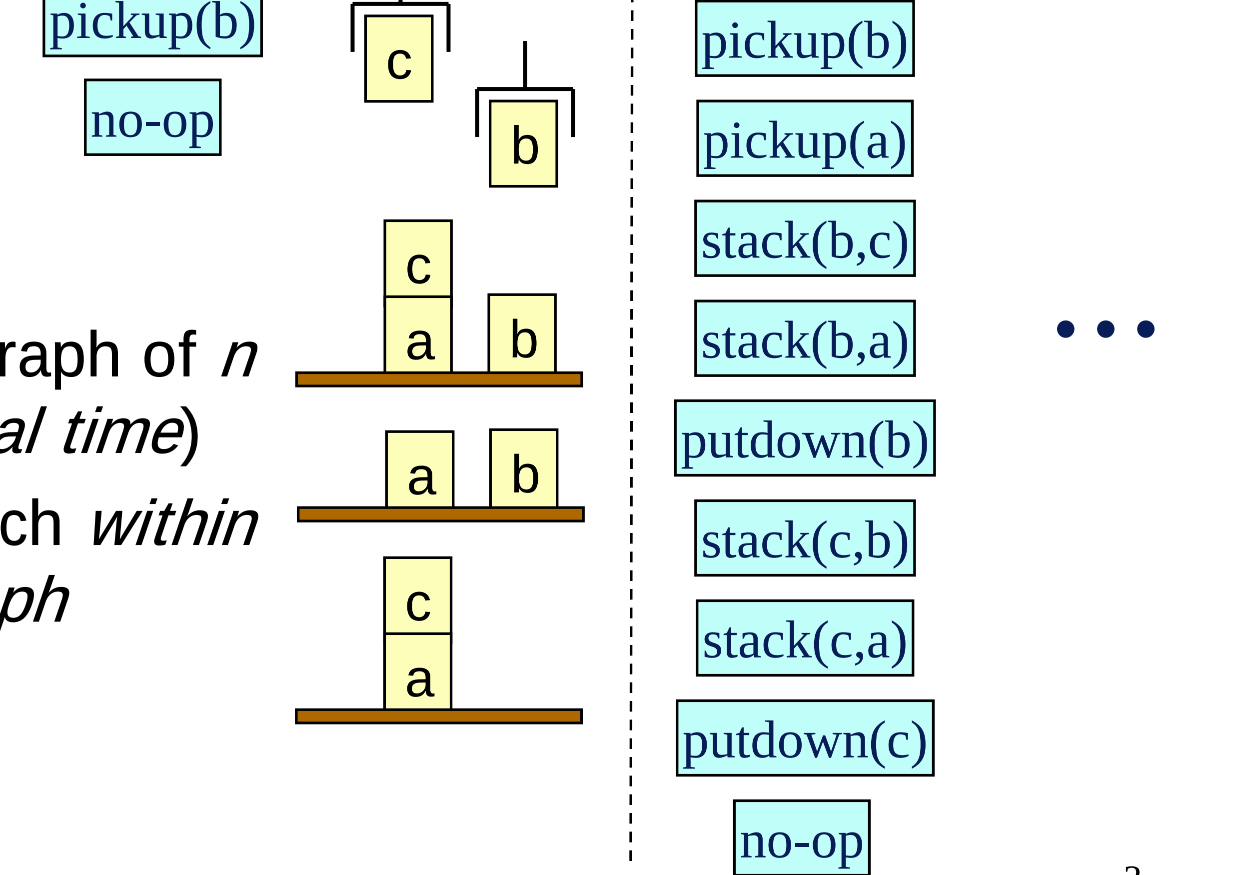
Homework

- Reading: S. Lee-Urban, et al., Transfer Learning of Hierarchical Task-Network Planning Methods in a RTS Game, In Proceedings of ICAPS 2007 Workshop on Planning and Learning (AIPL)

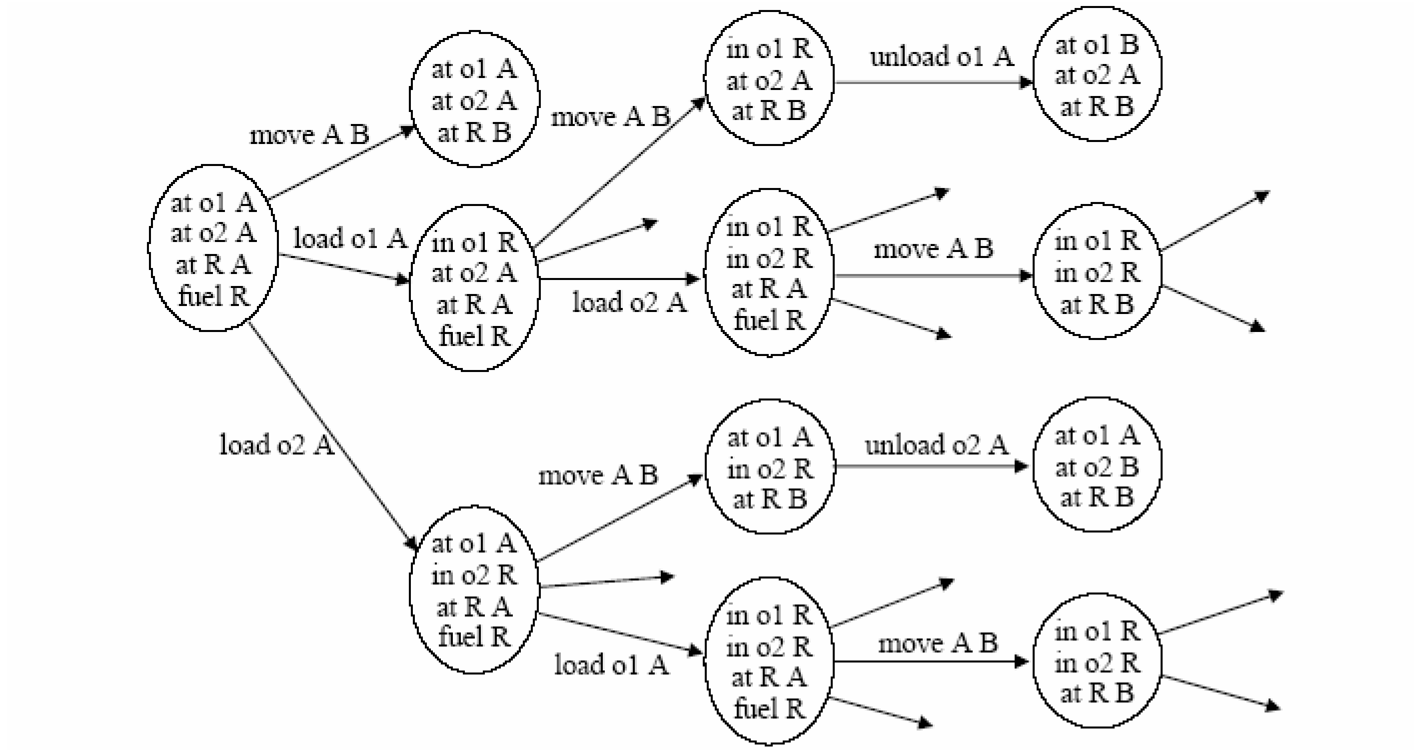


Graphplan

- For n = 1, 2, ...
  - Make planning graph of n levels (*polynomial time*)
  - State-space search *within the planning graph*



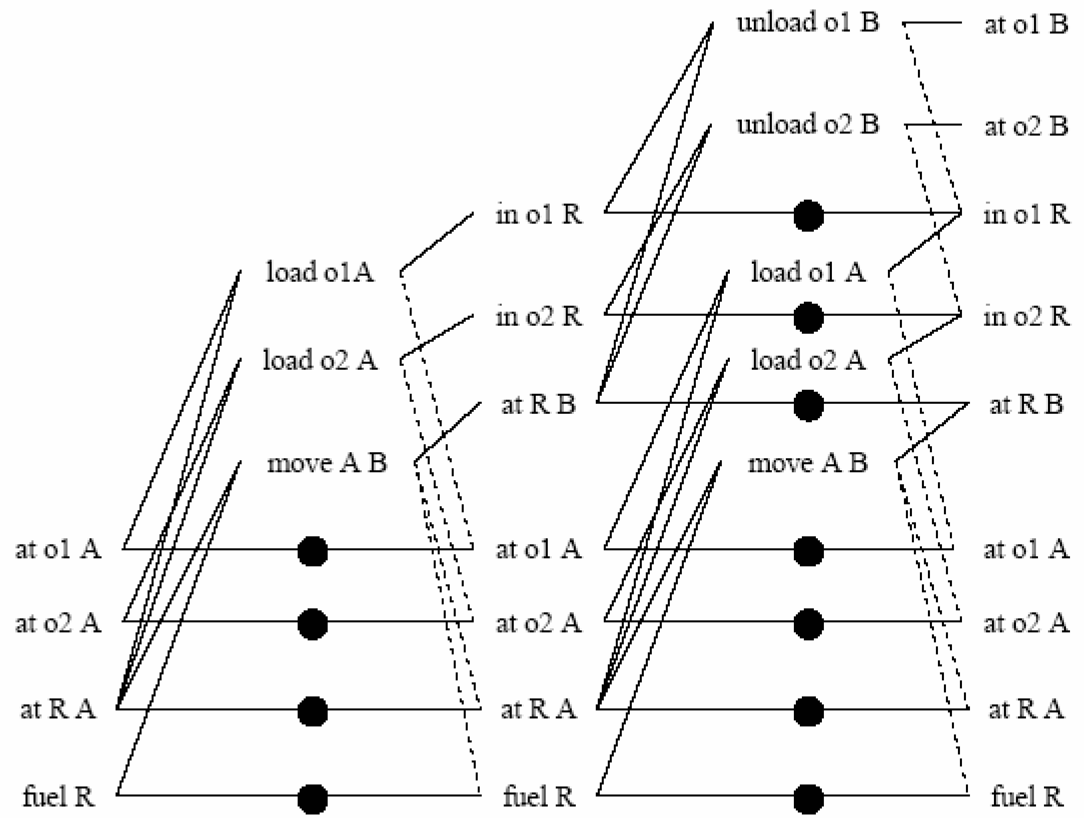
State-Space Planner



Graphplan

- Doing a reachability analysis for each goal
- Factors state-space into propositions and actions
  - To create ***action-level*** *i*:
    - Add each instantiated operator for which preconditions are all present at the previous proposition-level
    - Add all the no-op actions
  - To create ***proposition-level*** *i*+1:
    - Add all effects of the actions at action-level *i*
    - Distinguish add and delete effects
- Accounts for potential parallelism

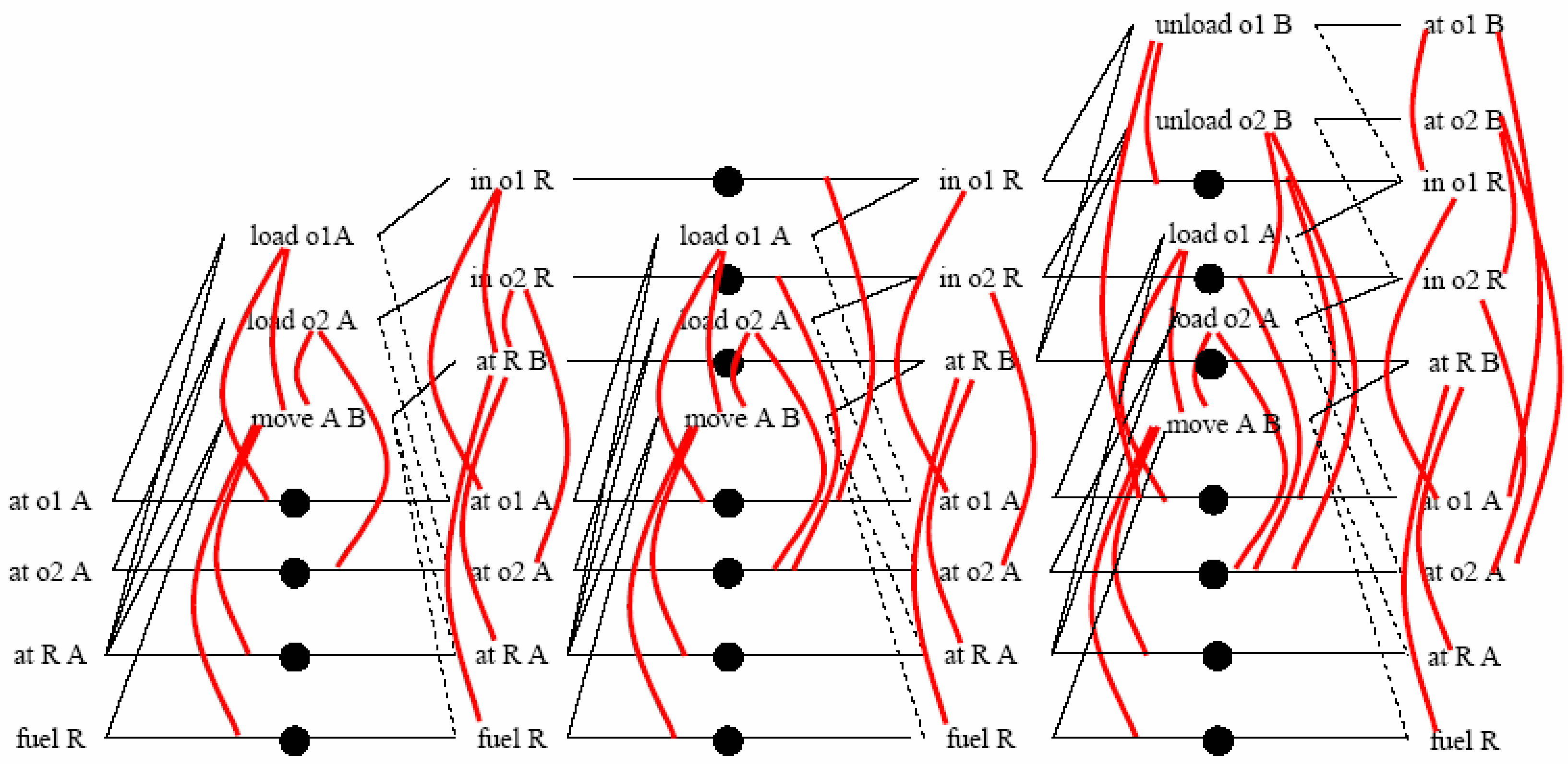
Naïve Plan Graph



Mutual Exclusivity Constraints

- Actions A and B are *exclusive,* at action-level *i*, if:
  - Interference: A (or B) deletes a precondition or an add-effect of B (or A)
  - Competing Needs: *p* is a precondition of A and *q* is a precondition of B, and *p* and *q* are exclusive in proposition-level *i* – 1
- Propositions *p* and *q* are *exclusive* in a proposition-level if:
  - ***All*** actions that add *p* are exclusive of ***all*** actions that add *q*

GraphPlan



unload o1 B
at o1 B
unload o2 B
at o2 B
in o1 R
in o2 R
load o1A
load o1 A
load o2 A
at R B
move A B
at o1 A
at o2 A
at R A
fuel R

What is good/bad about today's paper?

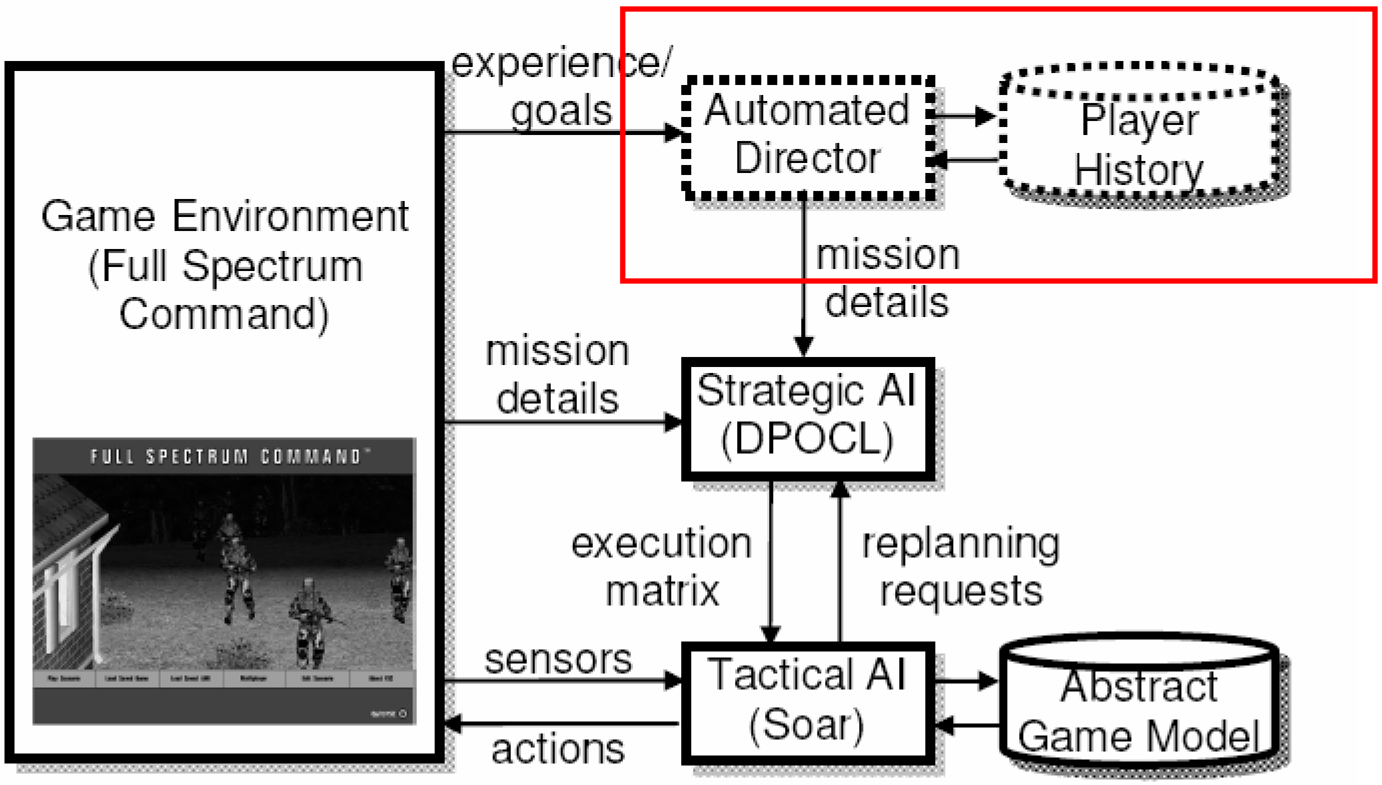
Full Spectrum Command

- Full Spectrum Command is a squad level game developed as a training tool for Army commanders engaged in MOUT
- Trying to train soldiers and commanders to be flexible to adapt to a broad range of scenarios
- Full Spectrum Warrior is a simpler commercial version of the game.
- FSC includes 3 game phases:
  - Planning
  - Execution
  - After-action Review

Full Spectrum Command

Adaptive Opponent Architecture

Future Work: using player history over multiple sessions



How can planning improve game-play?

How can planning improve game-play?

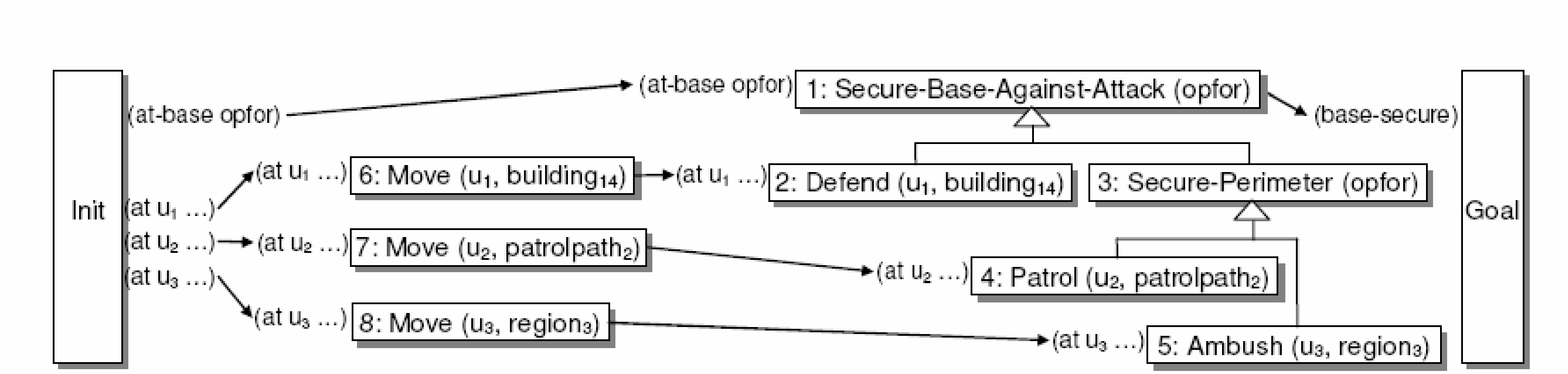
- Creation of opponents with multiple strategies (enhanced replayability)
  - Planners can be initialized with different world states, goals, and operators
  - Complete planners can find every possible solution plan for achieving a goal giving synthetic character the largest number of potential action choices.
- Why is replayability important?
- Other advantages:
  - Use of incompletely specified plans
  - Use of replanning

What kind of planner is DPOCL?

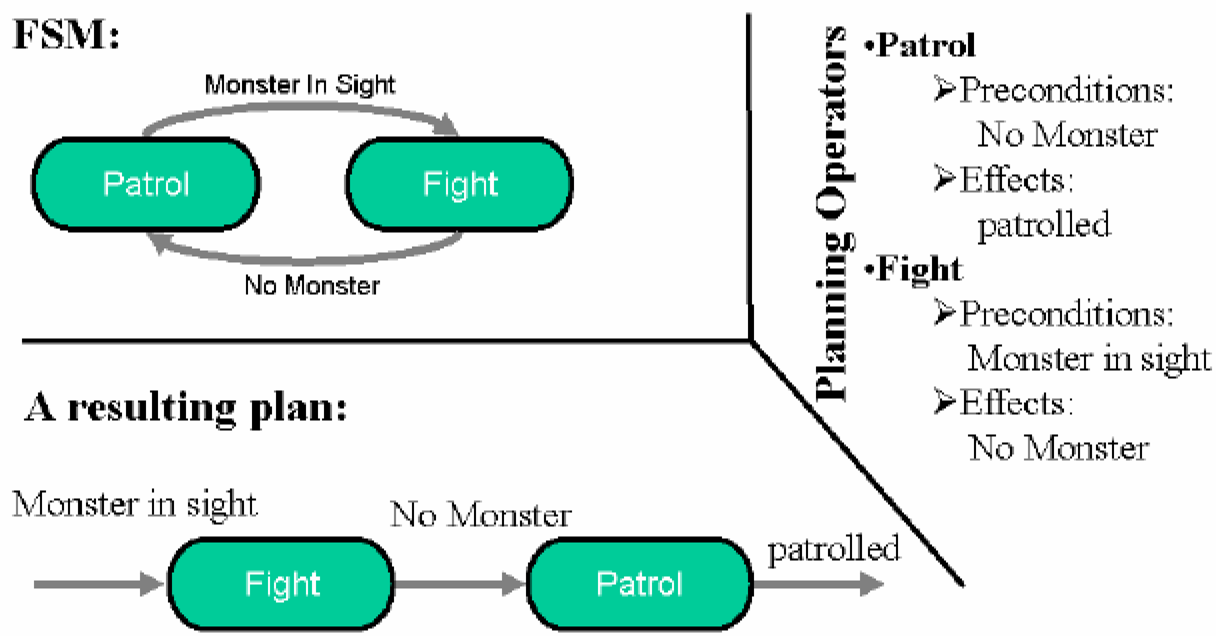
What kind of planner is DPOCL?

- Decompositional Partial Order Causal Link
- Hybrid between a plan-space planner (like UCPOP) and a hierarchical task network planner
- Partial order planner allows parallelism
- Backward chain from goal conditions by fulfilling preconditions of necessary operators
- Contains hierarchical decompositions of abstract operators (like an HTN planner)
- Unlike HTN planner, planning algorithm is applied recursively
- Plans are guaranteed to be sound assuming no uncertainty

DPOCL Planner

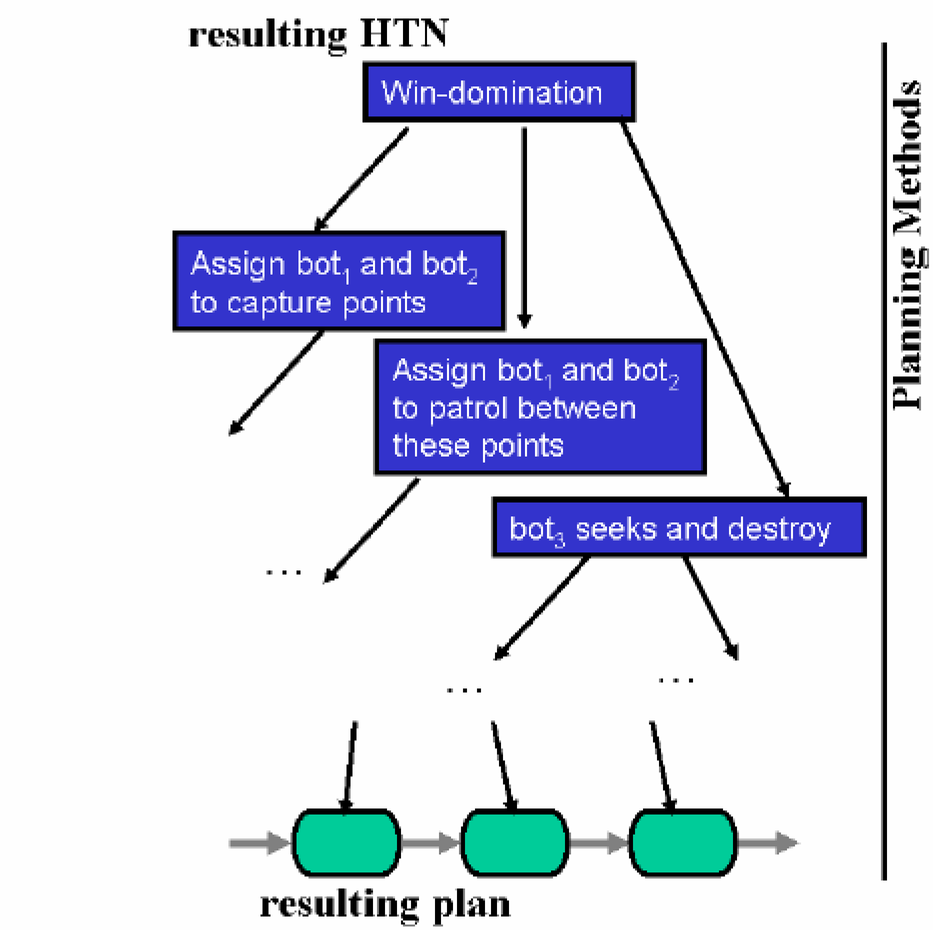


Finite State Machine



What are the advantages of a finite state machine?

Hierarchical Task Network



- **Control All Points**
  - Task:
    win-domination
  - Preconditions:
    - The team consists at least of 2 members
  - Subtasks:
    - Capture all domination points
    - Assign 2 members to patrol between those points
    - Assign remaining team to search and destroy task

Results

- 30 unique plans under 5 minutes
- Plans are evaluated using a heuristic taking into account
  - Optimality
  - Effective use of unit capabilities
  - Similarity to previous game session
  - Pedagogical/entertainment objectives (future work)
- Possible extensions:
  - Use of path-planning to improve heuristic
  - Use of contingency plans to avoid the computational cost of replanning

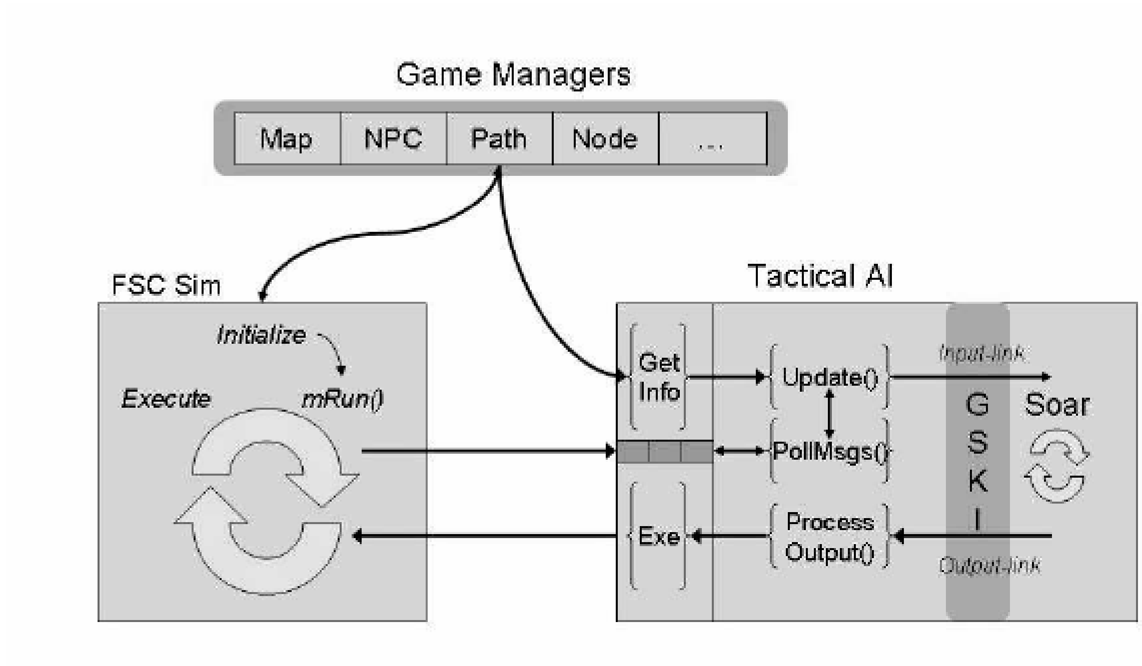
Tactical AI

- Planner outputs a strategy in the form of an execution matrix listing the proposed action for each unit at every time step
- How are these actions implmented in the game?
  - C++ object based execution system
    - Reliable, lacks variability
    - Probably FSM based
  - SOAR-based tactical AI execution library
    - SOAR rules can fire at any time in response to simulation events
    - Single SOAR instantiation controls a group of units and maintains a separate external goal stack

SOAR

- Stands for **S**tate, **O**perator, **A**nd **R**esult
- URL: http://sitemaker.umich.edu/soar/home
- Developed from Newell and Simon's General Problem Solver (GPS)
- Original purpose: to create a cognitive architecture that could integrate both goal-driven and reactive behavior
- Now: mainly used as a planning/execution system for simulated agents (especially in military simulation applications)
- What's the difference between cognitive architecture and any other type of planning system?

Tactical AI


Game Managers
Map
NPC
Path
Node
...
FSC Sim
Initialize
Execute
mRun()
Tactical AI
Get Info
Update()
PollMsgs()
Exe
Process Output()
Input-link
Output-link
G S K I
Soar

Proposed Evaluation

- Compare pre-existing game-industry AI vs. Tactical AI system
  - Computational and memory efficiency
  - Development time
  - Variability of behavior
  - Ease of extension
- Compare Strategic AI to plans generated by human players and mission designers