

---

# Metaplanning for Multiple Agents

Jonathan Gratch

Information Sciences Institute  
University of Southern California

June 7, 1998

# Overview

---

- ◆ Motivation:
  - support planning in dynamic multi-agent worlds
- ◆ Approach:
  - draw on classical and “intentional” planning
  - seamlessly combine:
    - » planning about actions
    - » reasoning about multiple plans
- ◆ Application to large-scale military simulations
- ◆ Issues and Future work

# Motivation

---

- ◆ Multi-agent planning
  - plans need collaborative generation and execution
  - plans must incorporate adversarial reasoning
- ◆ Dynamic planning
  - agents exist for extended time periods
  - plans and goals may change frequently over time
  - the environment may change unexpectedly
  - actions have duration and may fail

# Current Approaches

---

- ◆ Classical Approaches - *Noah, Ucpop, Strips,...*
  - Designed for static single-agent planning
  - Have been extended to more complex situations (Wilkins & Myers, Knoblock, Rickel, ...)
    - » interleave planning and execution
    - » handle uncertainty, conditional effects, conditional plans
    - » support multi-agent planning.
  - Collaboration typically tacked on
    - » e.g. Ensuring common knowledge, commitments

# Current Approaches

---

- ◆ “Intentional Planning”: shared plans, joint intentions, ERMA
- ◆ Richer view of planning process
  - Reason about intentional stances towards plans
    - » intend to, intend that, joint commitments
  - Support multiple plans: mine, yours, ours, theirs
  - Explicit support for collaboration
    - » contracting, coordinating actions, ...
  - Have tended to be normative theories
  - Unclear connection to classical approaches

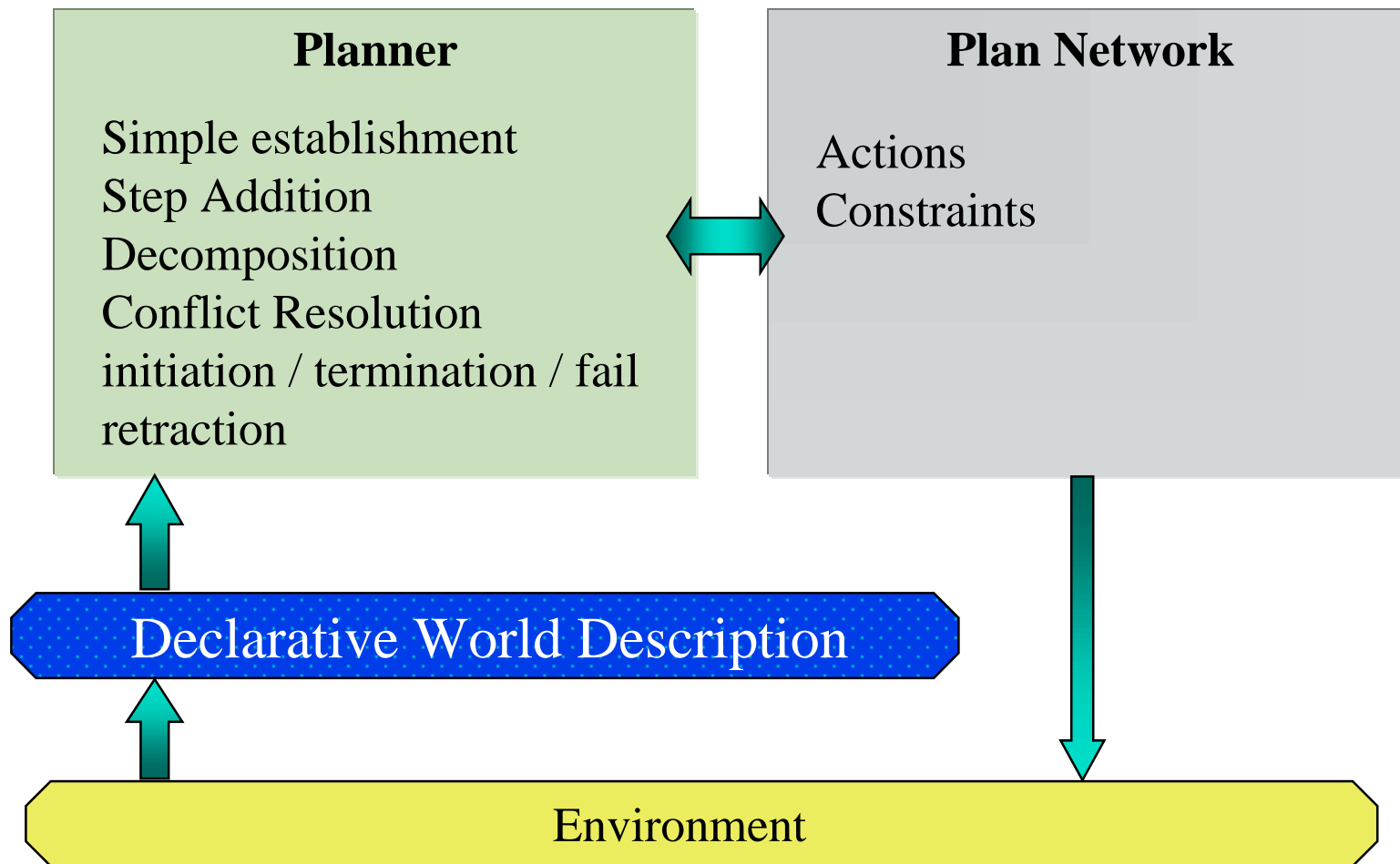
# Combined approach

---

- ◆ Draw on ideas from both approaches
- ◆ Integrate planning and intentional reasoning into single planning algorithm
- ◆ Try to clarify the connections:
  - From classical perspective”
    - » incorporate “intentional” reasoning
  - From intentional perspective
    - » ground semantics of intentions in the operations classical planning algorithm

# Planner: based on IPEM / X11

---



# Action Language

---

## **defTask**

```
Receipt_of_Mission (?recipient ?sender ?order ?suborder ?plan) {
:pre ( (p-op:  order(?sender ?recipient ?order)))
:add ( (a-sup:  suborder(?order ?recipient ?suborder))
      (a-ord:  order(?recipient ?recipient ?suborder))
      (a-plfr:  plan-for(?recipient ?suborder ?plan))
      (a-plan:  plan(?plan))
      (a-stat:  plan-status(?plan UNAPPROVED)))
:bindings ((?recipient ≠ ?sender) (?order ≠ ?suborder))
:commands ( ;# when-added | at-start | at-end | at-failure
           (:at-start ?plan = create-plan())
           (:at-start ?suborder = extract-order(?recipient ?order))
           (:at-start populate-plan(?plan ?suborder))
           (:at-start disable-modification(?plan)))
}
```

# Decomposition / Specialization

---

```
defRefinement PassLinesOut {  
  :task (Move(?G ?ST ?S-CM ?END ?E-CM ?RT ?FORM ?SP ?SF ?CMS))  
  :conditions (  
    (:filter flot(?FLOT) :at-start step2)  
    (:test crosses-flot(?RT) == YES)  
    (:filter flot-status(?G BEFORE) :at-start step2))  
  :expansion (  
    (step1: Move(?G ?ST ?S-CM ?END1 ?PP ?HEAD ?F1 ?SP1 ?SF ?CM1))  
    (step2: PASS_LINES_OUTBOUND(?G ?PP))  
    (step3: Move(?G ?END1 ?PP ?END ?E-CM ?TAIL ?F2 ?SP2 ?SF ?CM2)))  
  :links (  
    (step1:a-at == step3:p-at)  
    (step1:a-atcm == step3:p-atcm))  
  :orderings ((step1 < step2) (step2 < step3))  
  :commands (  
    (:when-added ?PP = passage-point(?RT ?FLOT))  
    (:when-added ?HEAD = route-head(?RT ?PP))  
    (:when-added ?TAIL = route-tail(?RT ?PP))  
    (:when-added ?END1 = end-point(?PP)))  
}
```

# 4 ideas: 1) Multiple Plans

---

- ◆ Problem:
  - Want to reason about multiple plans
  - Want to recognize inter-plan interactions
- ◆ Redefine “plan”
  - Distinguish between plan and “plan network”
  - A plan is some subset of the plan network
  - compute “inter-plan” threats for free
- ◆ Allow plan properties and plan relations
  - hypothetical, intended, executable, flawed, ...

## 2) Modulating planning

---

- ◆ Problem:
  - Different plans must be treated differently
    - » mine vs. yours
    - » hypothetical vs. intended vs. executing
- ◆ Plan properties change the planner's behavior w.r.t. elements of that plan
  - can't initiate actions in an unexecutable plan
  - can't repair an unmodifiable plan
    - » unless we deliberately make it modifiable

# 3) Grounding Intentions

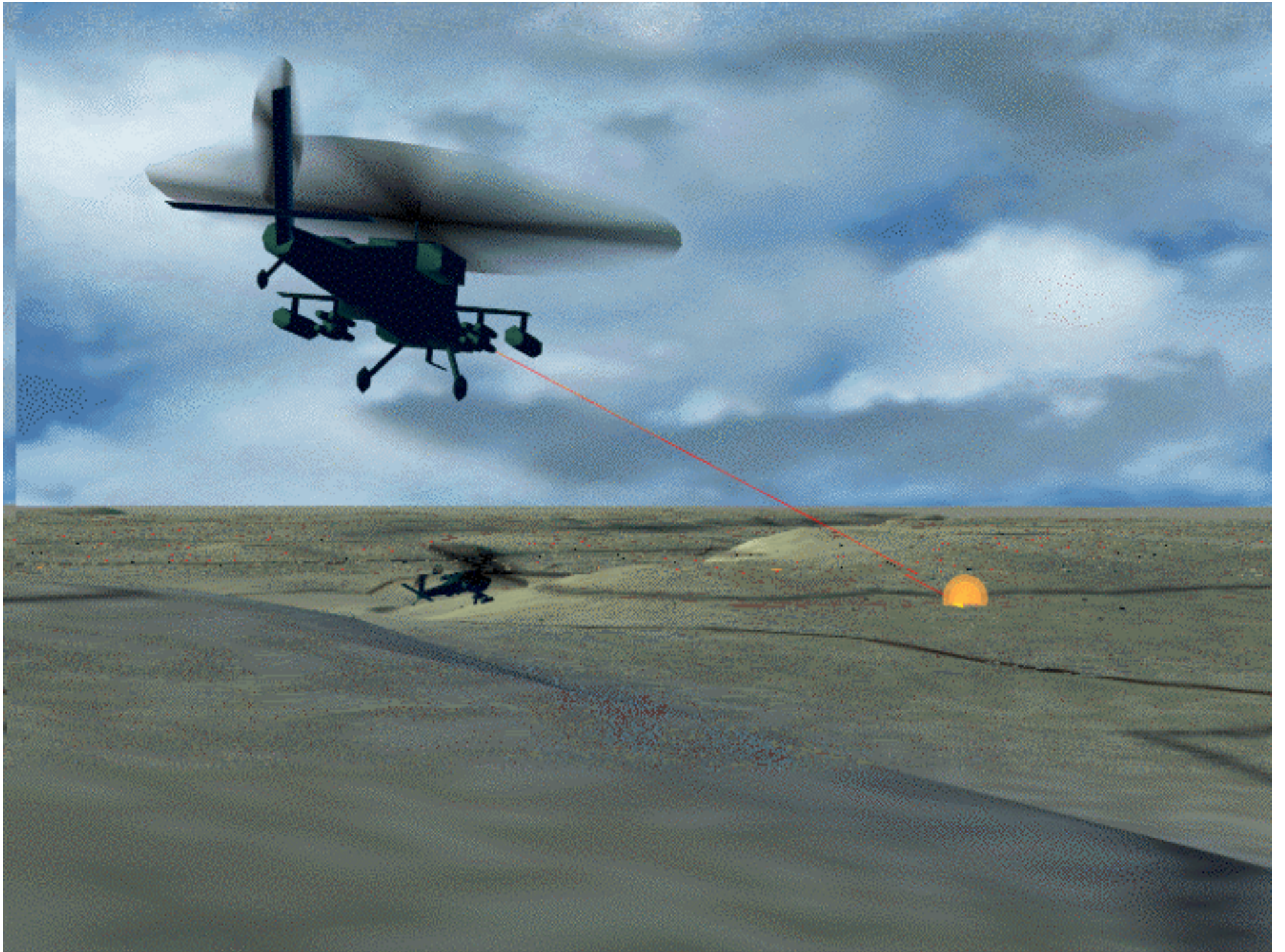
---

- ◆ Problem:
  - Connection between planning and “intentions”?
- ◆ Intentions modulate planning behavior via plan properties / relations
  - e.g. “Intends that” - Grosz and Kraus
    - interpret as a statement about handling inter-plan threats
      - planner avoids introducing threats into other’s plans
      - planner introduces actions that resolve other’s threats
  - For adversarial reasoning:
    - planner introduces threats into other’s plan

## 4) Metaplanning

---

- ◆ Problem:
  - Don't want separate reasoner for intentions
- ◆ This reasoning can be represented as plans
  - planner already supports multiple plans
  - make one plan a “intentional” (meta) plan
  - executing meta-actions results in formation of intentions that modulate behavior of planner w.r.t. other plans



# Tactical simulations for training

---

- ◆ Battalion-level deep-strike missions
  - 1 battalion planning agent
  - 2 company planning agents
  - 10 helicopter execution agents
  - several hundred other friendly and enemy units
  - Participated in 2-day simulated exercise: STOW97
- ◆ Collaborative planning and execution in hierarchical organizations
  - develop plan, contract out details to subordinates, monitor execution and replan as needed

Refresh

Plan a step

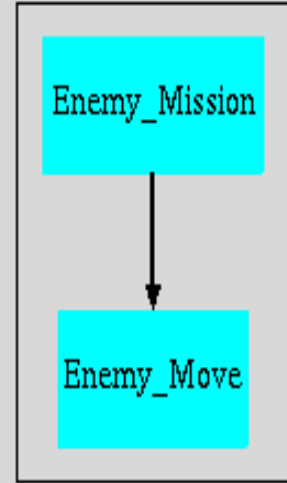
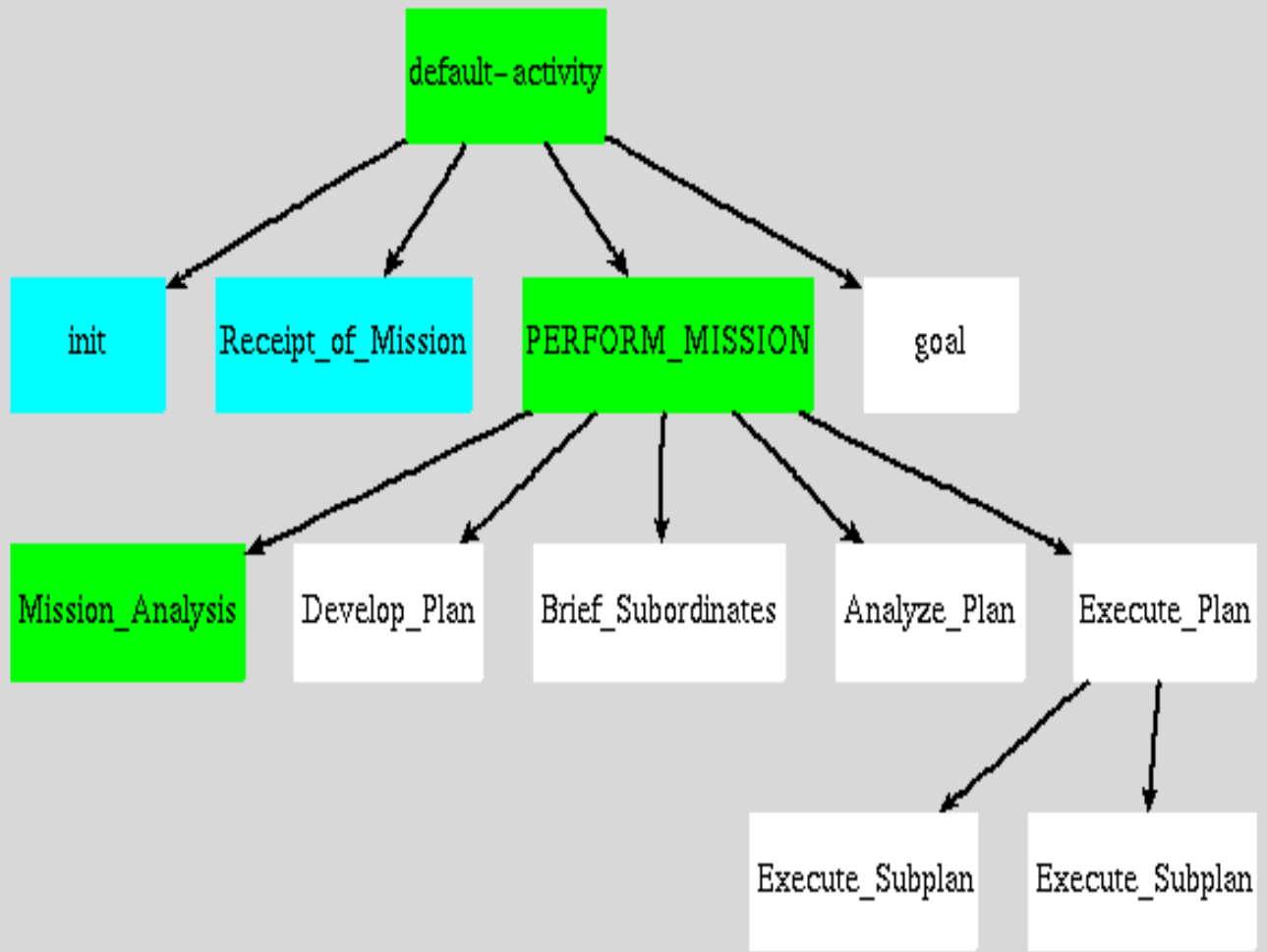
Hide

Quit

Default

multi\_16bn

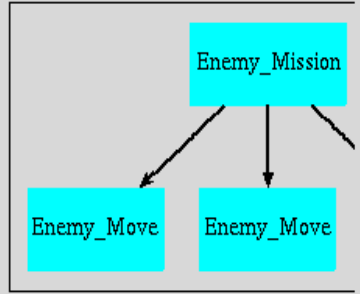
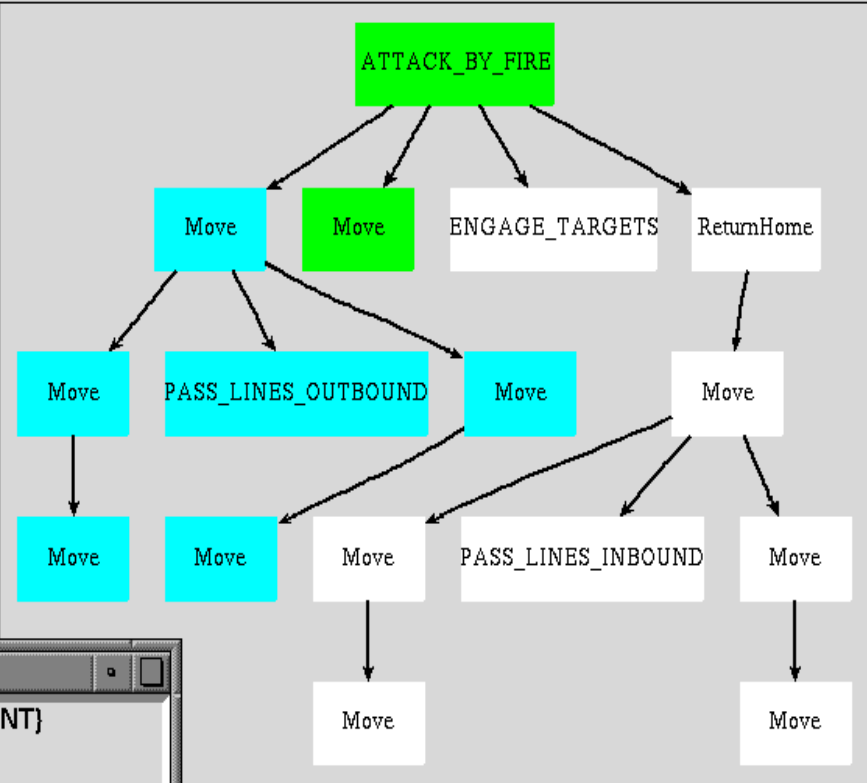
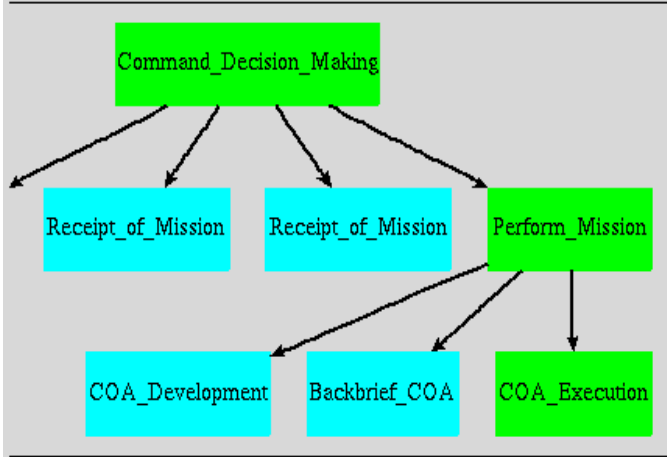
Enemy\_Mission



planning-operator handle-unexpected-effect

multi\_16bn\_16bco

Enemy\_Mission



**PASS\_LINES\_OUTBOUND**

```

PASS_LINES_OUTBOUND {?GROUP ?PASSAGE-POINT}

:pre {{p-bef: flot-status{?GROUP BEFORE}}}
:add {{a-aft: flot-status{?GROUP AFTER}}}
:del {{d-bef: flot-status{?GROUP BEFORE}}}
:commands {
  {:at-start coordinate-passage{?GROUP ?PASSAGE-POINT}}}
    
```

State:           executed  
 Task-id:        S127  
 GROUP:          I130 (16bco)  
 PASSAGE-POINT: P296 (pp-16bco-route152)

OK

124: initiate-task: Move (T32)

Mode: continuous

# Issues

---

- ◆ Need more theoretical commitments
  - Provides a platform for flexible reasoning but:
    - Use domain specific search control for:
      - » balance planning and execution
      - » respond to changes in the world
    - Use domain theory to specify:
      - » intentional reasoning
      - » how to maintain coordination

# Issues

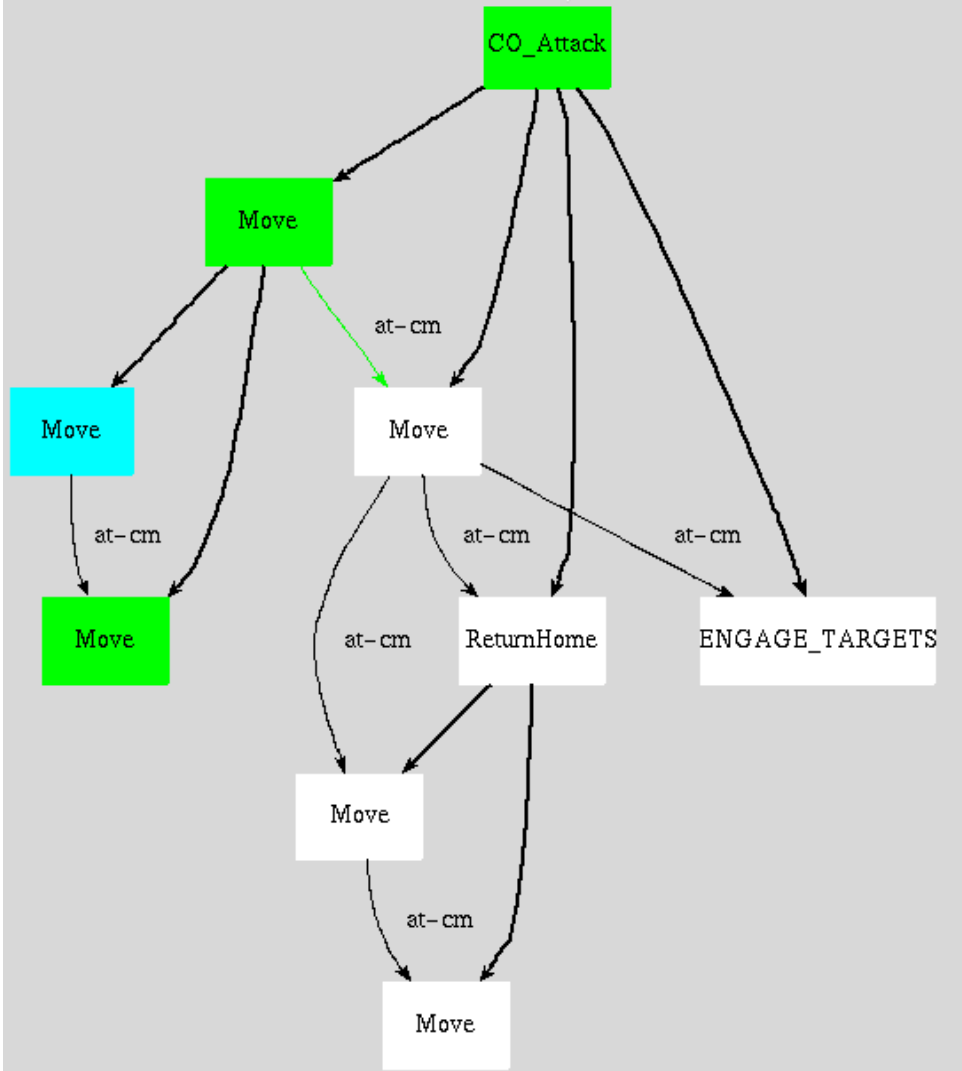
---

- ◆ Collaboration in hierarchical organizations
  - What is a primitive task?
  - Different levels have different domain theories
    - » have to resolve ambiguities
    - » have to resolve conflicting views
  - Plan execution involves plan recognition of subordinate activities

# Issues

---

- ◆ Theoretical analysis
  - what is the relation to Shared Plans, etc.
  - clarify plan semantics (what belongs in a plan)
- ◆ Planning
  - iterative repair via validation structure
  - control of search



Step "Move"

```

Move {?GROUP ?S-CM ?E-CM ?ROUTE ?CMS}

:actual-name PERFORM_TACTICAL_MOVEMENT
:pre {{p-atcm: at-cm{?GROUP ?S-CM}}}
:add {{a-atcm: at-cm{?GROUP ?E-CM}}}
:del {{d-atcm: at-cm{?GROUP ?S-CM}}}
:bindings {{?S-CM != ?E-CM}}
:commands {{:at-start resume{?GROUP ?E-CM}}}
  
```

Plan: cluster\_multi\_16bn\_16bco  
 State: executed  
 Task-id: T416  
 GROUP: I125 (16bco)  
 S-CM: S158 (start-16bco)  
 E-CM: N377 (pp-16bco-route174)  
 ROUTE: N306  
 CMS: C346

OK

111: initiate-task: Move (S201)

Mode: continuous