AI Planning

Daniel S. Weld
University of Washington

1

Outline

- Agent Control & Classical Planning
  - Reactive *vs.* Planned Control
  - Generative Planning
  - STRIPS & Assumptions of Classical Planning
- Planning as Search
- Graphplan
- Uncertainty & Expressive Action Languages

2

2 Approaches to Agent Control

- ***Reactive Control***
  - Set of situation-action rules
  - E.g.
    - 1) **if** dog-is-behind-me
      then run-forward
    - 2) **if** food-is-near
      then eat
- ***Planning***
  - Reason about effect of combinations of actions
  - "Planning ahead"
  - Avoiding "painting oneself into a corner"

3

Different Planning Approaches

- Generative Planning
  - Reason from first principles (knowledge of actions) to generate plan
  - Requires formal model of actions
- Case-Based Planning
  - Retrieve old plan which worked for similar problem
  - Revise retrieved plan for this problem
- See also
  - Policy Iteration / Markov-Decision Processes
  - Reinforcement Learning

4

Generative Planning

- Input
  - Description of initial state of world *(in some KR)*
  - Description of goal *(in some KR)*
  - Description of available actions *(in some KR)*
- Output
  - Sequence of actions

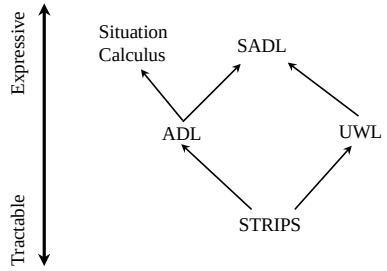
5

Input Representation

- *Description of initial state of world*
  - Set of propositions:
  - **((block a) (block b) (block c) (on-table a) (on-table b) (clear a) (clear b) (clear c) (arm-empty))**
- *Description of goal (i.e. set of desired worlds)*
  - Logical conjunction
  - Any world that satisfies the conjunction is a goal
  - **(and (on a b) (on b c)))**
- *Description of available actions*

6

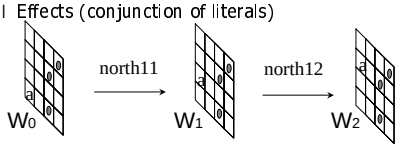
Representing Actions



7

How Represent Actions?

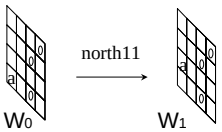
- ***Simplifying assumptions***
  - Atomic time
  - Agent is omniscient (no sensing necessary).
  - Agent is sole cause of change
  - Actions have deterministic effects
- ***STRIPS representation***
  - World = set of true propositions
  - Actions:
    - Precondition: (conjunction of literals)
    - Effects (conjunction of literals)



8

STRIPS Actions

- Action =function from world-state→world-state
- Precondition says where function defined
- Effects say how to change set of propositions



Note: strips doesn't allow derived effects; you must be complete!

north11
precond: (and (agent-at 1 1) (agent-facing north))
effect: (and (agent-at 1 2) (not (agent-at 1 1)))

9

Action Schemata

- Instead of defining:
  pickup-A and **pickup-B** and …
- Define a schema:

(:operator **pick-up**
:***parameters*** **((block ?ob1))**
:*precondition* (and (clear ?ob1)
(on-table ?ob1)
(arm-empty))
:*effect* (and (not (clear ?ob1))
(not (on-table ?ob1))
(not (arm-empty))
(holding ?ob1)))

Note: strips doesn't allow derived effects; you must be complete!

10

Outline

- Agent Control & Classical Planning
- Planning as Search
  - Forward chaining
  - Backward chaining
  - Search thru the space of partial plans
- Graphplan
- Uncertainty & Expressive Action Languages

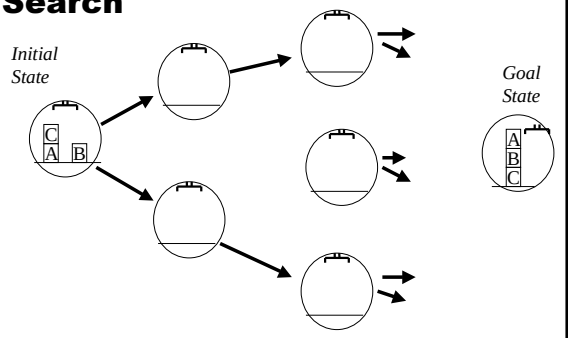
11

Planning as Search

- Nodes
  World states
- Arcs
  Actions
- Initial State
  The state satisfying the complete description of the initial conds
- Goal State
  Any state satisfying the goal propositions

12

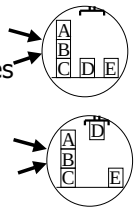
Forward-Chaining World-Space Search



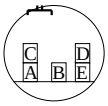
13

Backward-Chaining Search Thru Space of Partial World-States

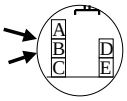
- Problem: Many possible goal states are equally acceptable.
- From which one does one search?



Initial State is completely defined



* * *



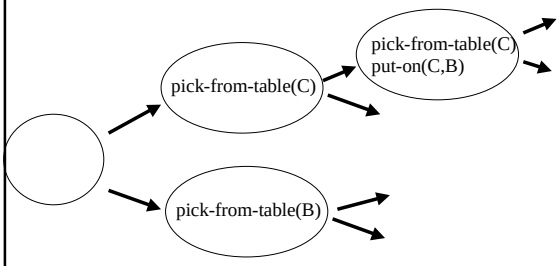
14

"Causal Link" Planning

- Nodes
  - Partially specified plans
- Arcs
  - Adding + deleting actions or constraints (e.g. <) to plan
- Initial State
  - The empty plan
  - (Actually two dummy actions…)
- Goal State
  - A plan which when simulated achieves the goal
  - Need efficient way to evaluate quality (percentage of preconditions satisfied) of partial plan …
  - Hence causal link datastructures

15

Plan-Space Search



- How represent plans?
- How test if plan is a solution?

16

Outline

- Agent Control & Classical Planning
- Planning as Search
- Graphplan
  - The planning graph
  - Mutual exclusion
  - Solution extraction
  - Viewed as constraint satisfaction
  - Compilation to SAT
- Uncertainty & Expressive Action Languages

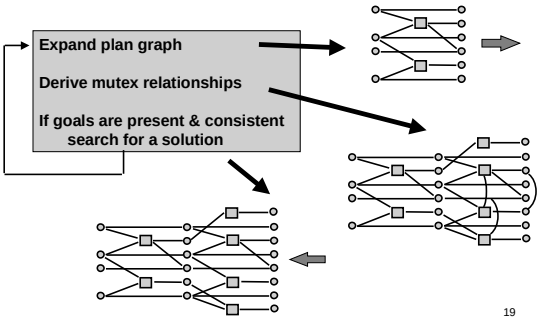
17

Planning as Search 3 Graphplan

- Phase 1 - Graph Expansion
  - **Necessary (insufficient) conditions for plan existence**
  - **Local consistency of plan-as-CSP**
- Phase 2 - Solution Extraction
  - **Variables**
    - **action execution at a time point**
  - **Constraints**
    - **goals, subgoals achieved**
    - **no side-effects between actions**

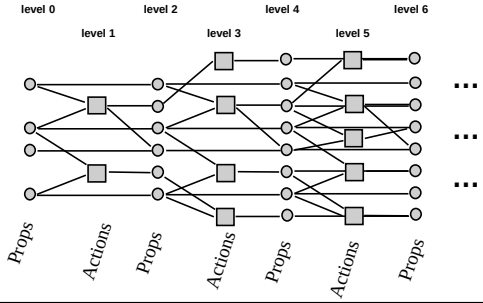
18

Graphplan



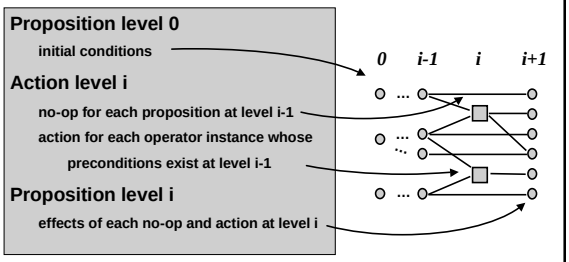
19

The Plan Graph



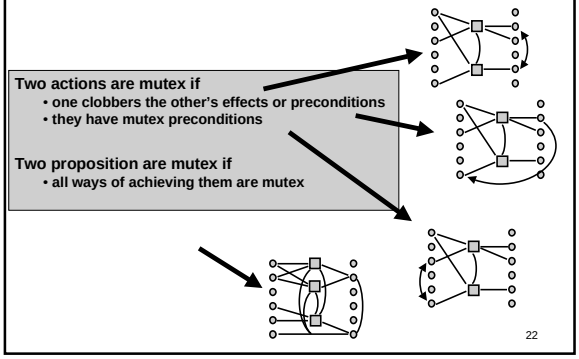
20

Graph Expansion



21

Mutual Exclusion



22

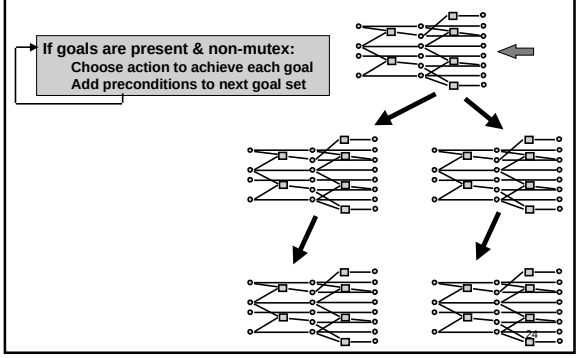
Graphplan

- Create level 0 in planning graph
- Loop
  - If goal $\subseteq$ contents of highest level (nonmutex)
  - Then search graph for solution
    - If find a solution then return and terminate
  - Else Extend graph one more level

A kind of double search: forward direction checks necessary (but insufficient) conditions for a solution, ... Backward search verifies...

23

Searching for a solution



24

Dinner Date

Initial Conditions: (:and (cleanHands) (quiet))

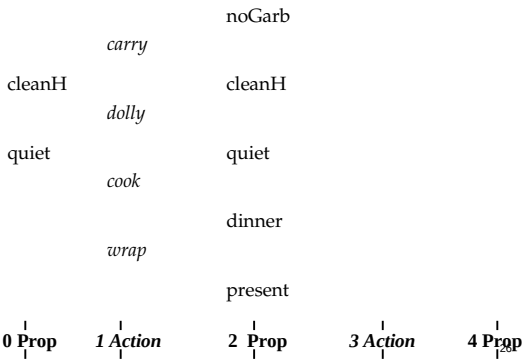
Goal: (:and (noGarbage) (dinner) (present))

Actions:

(:operator **carry** *:precondition*
:effect (:and (noGarbage) (:not (cleanHands)))
(:operator **dolly** *:precondition*
:effect (:and (noGarbage) (:not (quiet)))
(:operator **cook** *:precondition* (cleanHands)
:effect (dinner))
(:operator **wrap** *:precondition* (quiet)
:effect (present))

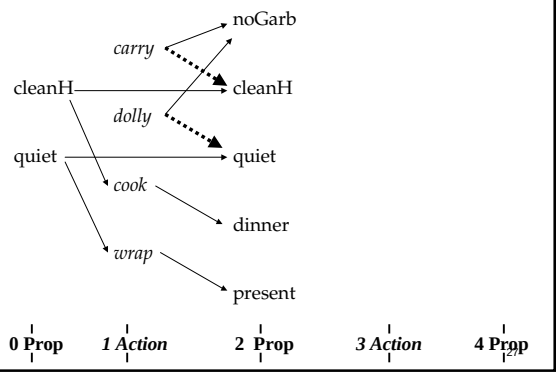
25

Planning Graph



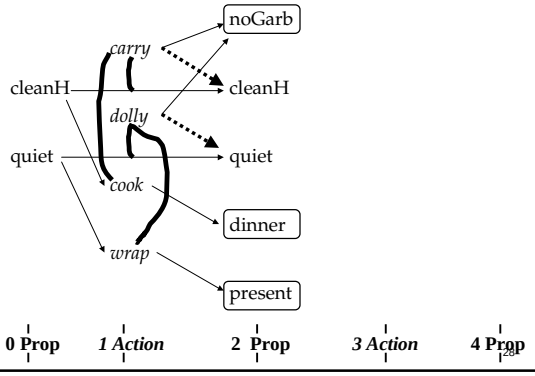
26

Are there any exclusions?



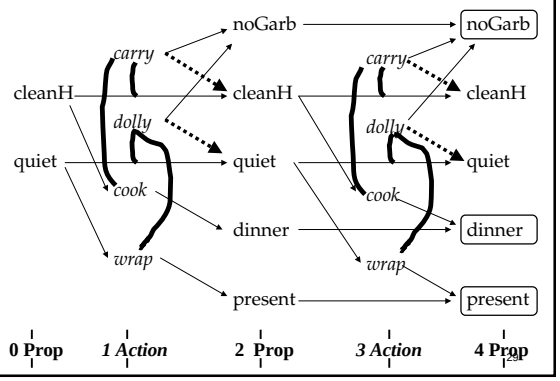
27

Do we have a solution?



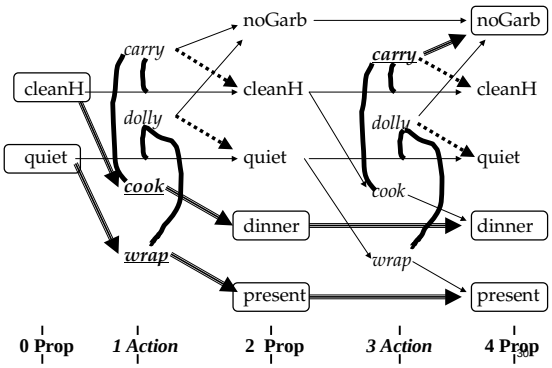
28

Extend the Planning Graph



29

One (of 4) possibilities



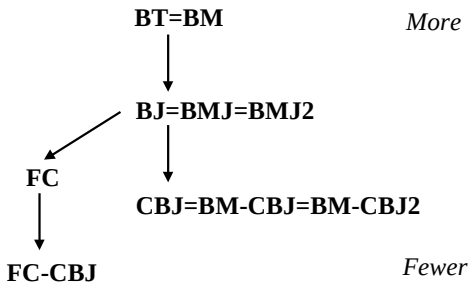
30

Constraint Satisfaction

- Chronological Backtracking (BT)
- Backjumping (BJ)
- Conflict-Directed Backjumping (CBJ)
- Forward checking (FC)
- Dynamic variable ordering heuristics
- Preprocessing Strategies

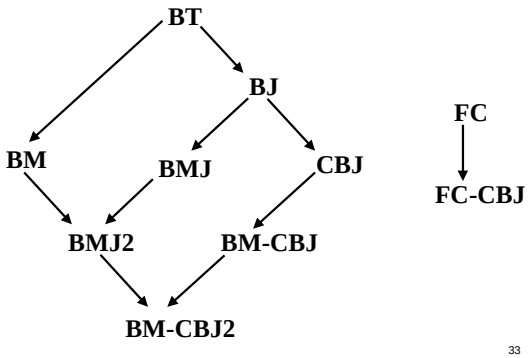
31

Number of Nodes Explored



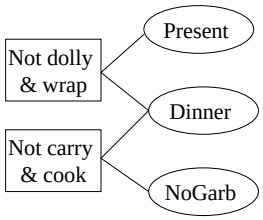
32

Number of Consistency Checks



33

Graphplan Solution Extraction as a Constraint Network



34

Dynamic variable ordering

- In the N-queens examples we assumed
  - First x_1 then x_2 then ...
- But this order not required
  - ***Any*** order ok with respect to completeness
  - A good order leads to huge speedup
- A good heuristic (MRV):
  - Choose variable w/ minimum remaining values
    - This is easy if one is doing FC

35

DVO MRV => WOW!!

Algo	17 Queens	21 Queens	27 Queens
$FC\text{-}CBJ_{mrv}$	1959	2572	5602
FC-CBJ	67090	114612	737008
FC	67329	115120	7448781
CBJ	428645	949128	
BJ	436340	972065	
BT	485597	1156015	

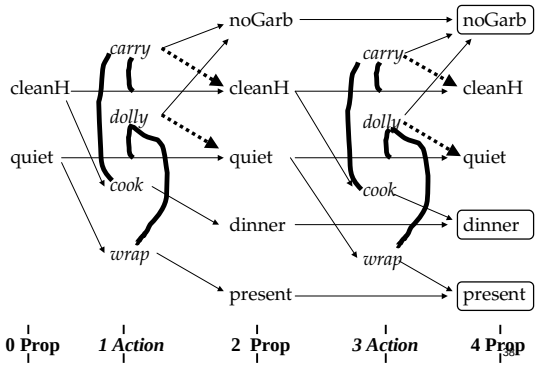
36

Preprocessing Strategies

- Even FC-CBJ is O(b^d) time worst case
- Sometimes useful to spend polynomial time preprocessing to achieve local consistency before doing exponential search
- Arc consistency
  - Consider all pairs of vars
  - Can any values be eliminated from a domain ala FC
  - Propagate
  - O(d^2) time where d= number of vars

37

Compilation to SAT



38

Axiom Schemata

- Init
- Goal
- Frame
  - Classical
  - Explanatory
- At Least One
- Exclusion
- No Partial

39

Outline

- Agent Control & Classical Planning
- Planning as Search
- Graphplan
- Uncertainty & Expressive Action Languages
  - Disjunction
  - Quantification
  - Conditional effects
  - Incomplete information

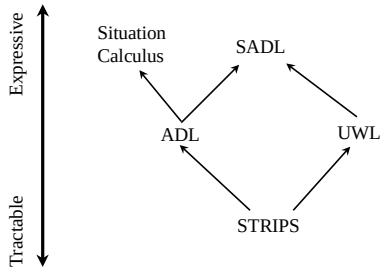
40

Generative Planning

- Input
  - Description of initial state of world ***(in some KR)***
  - Description of goal ***(in some KR)***
  - Description of available actions ***(in some KR)***
- Output
  - Sequence of actions

41

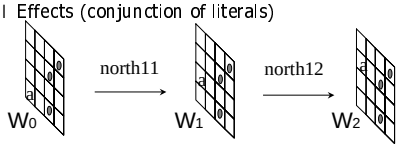
Representing Actions



42

How Represent Actions?

- ***Simplifying assumptions***
  - Atomic time
  - Agent is omniscient (no sensing necessary).
  - Agent is sole cause of change
  - Actions have deterministic effects
- ***STRIPS representation***
  - World = set of true propositions
  - Actions:
    - Precondition: (conjunction of literals)
    - Effects (conjunction of literals)



43

Graphplan$_{strips}$

- Create level 0 in planning graph
- Loop
  - If goal $\subseteq$ contents of highest level (nonmutex)
  - Then search graph for solution
    - Implement as a dynamic CSP in which:
      - Variables = (sub)goals at a level
      - Values = actions (at prev level) that might make goal true
      - Optimize with FC-CBJ and MRV variable-ordering heuristic
    - If find a solution then return and terminate
  - Else Extend graph one more level

44

ADL Action Representation

- STRIPS
  - Preconditions: propositional conjunctions
  - Effects: propositional conjunctions
- ADL
  - Preconditions:
    - Universal & Existential Quantification
    - Disjunction
  - Effects
    - Universal Quantification
    - Conditional Effects
  - Domain Axioms

45

Why ADL?

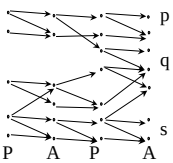
- Universally Quantified Preconditions
  - Blocks World Example
  - UNIX Example
- Conditional Effects
  - Light Switch
- Universally Quantified Conditional Effects
  - Spraypaint
  - UNIX example

46

Implementing Disjunction

- Convert precondition to DNF
- Add an action instance to graph if
  - Any disjunct is present (nonmutex) at prev layer

```
(defschema (foo)
        precondition: (and p (or q r s)
        effect:       (and t u))
```



47

Quantified Preconditions

Locations: Seattle, Renton
Vehicles: truck37, truck44, plane55

(exists (?l - location)
  (forall (?v - vehicle) (at ?v ?l)))

(forall (?l - location)
  (exists (?v - vehicle) (at ?v ?l)))

48

Preconditions vs Effects

- Disjunctive effects
- Existentially quantified effects

49

Conditional Effects

(defschema (flip)
:parameter (?l - light)
:precondition
:effect (when (on ?l) (not (on ?l)))
(when (not (on ?l)) (on ?l))))

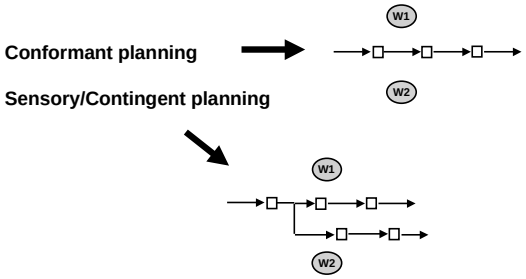
(defschema (flip-on)
:parameter (?l - light)
:precondition (not (on ?l))
:effect (on ?l))

(defschema (flip-off)
:parameter (?l - light)
:precondition (on ?l)
:effect (not (on ?l)))

50

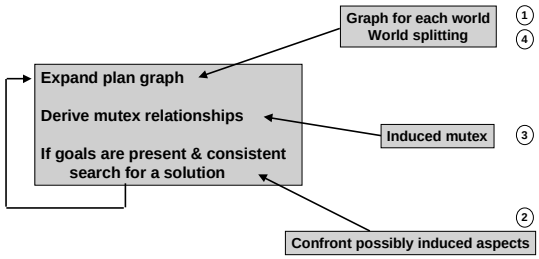
Incomplete Information

Conformant .vs. Contingent



51

Conformant Graphplan



52

The Little Lost Rover

Goal: Transfered

Inits:
¬Transfered
Blocked ⊗ Weak
¬Dead

Operators:
Transfer
pre: ¬Dead & ¬Blocked
eff: Transfered

Charge
eff: ¬Weak

Move
eff: ¬Weak & Blocked ⇒ ¬Blocked
Weak ⇒ Dead

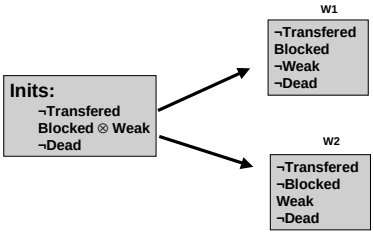
Aspects

Move1
pre: Blocked & ¬Weak
eff: ¬Blocked

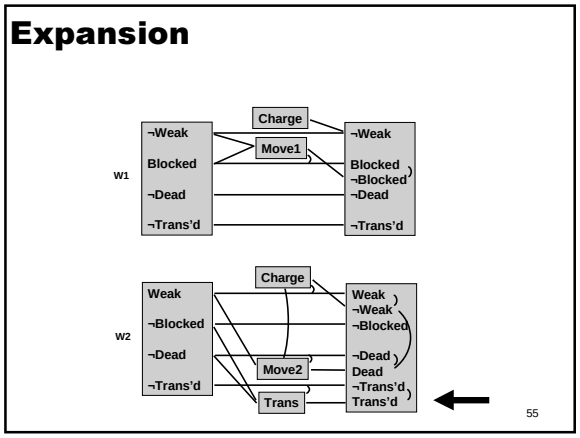
Move2
pre: Weak
eff: Dead

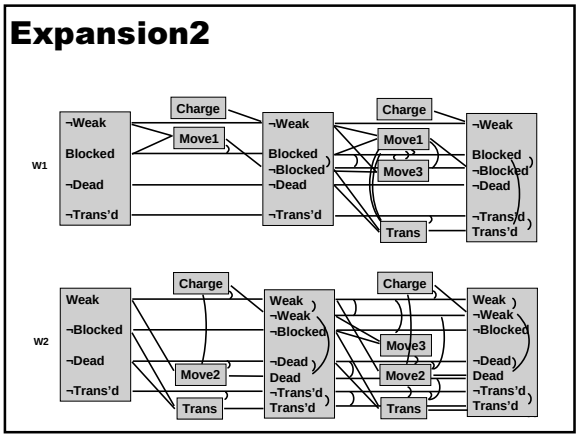
Move3
pre: ¬Blocked & ¬Weak
eff:

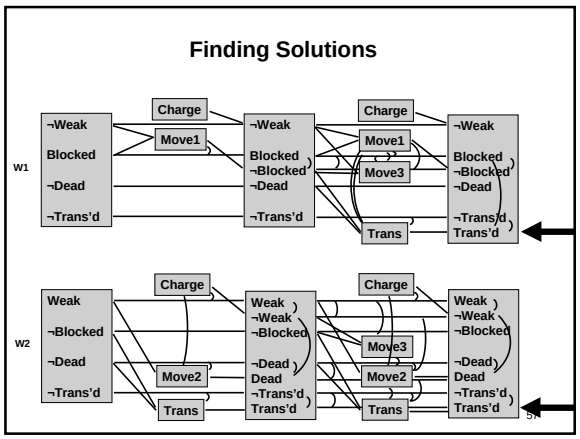
Possible Worlds

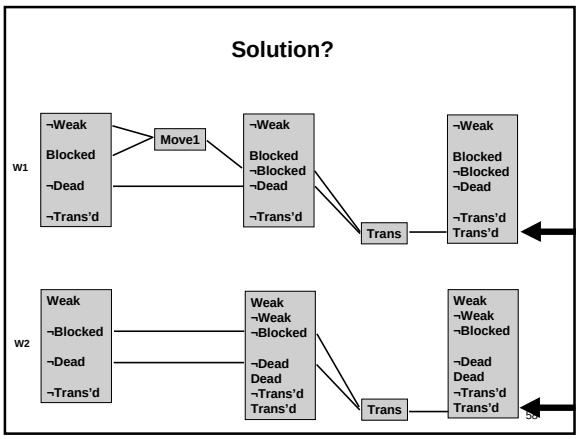


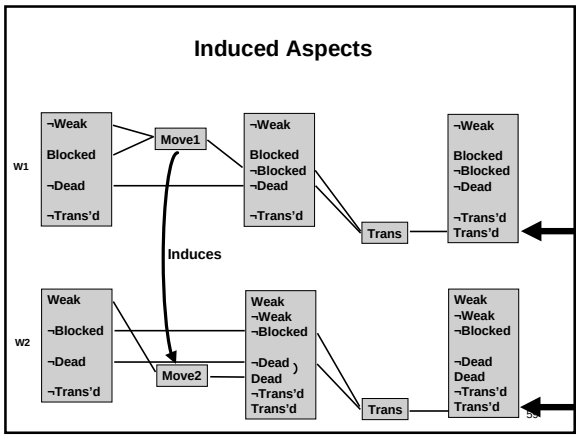
54

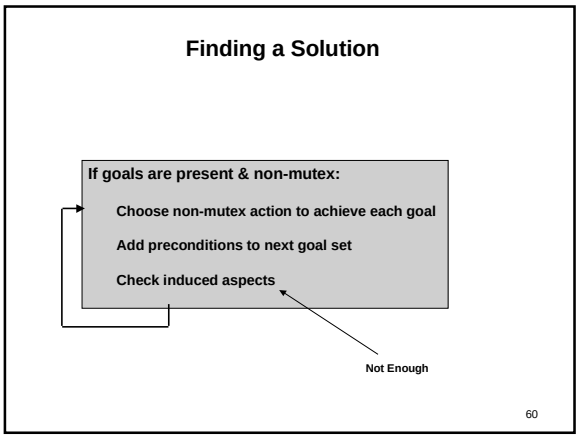

Expansion
W1
¬Weak
Blocked
¬Dead
¬Trans'd
Charge
Move1
¬Weak
Blocked
¬Blocked
¬Dead
¬Trans'd
W2
Weak
¬Blocked
¬Dead
¬Trans'd
Charge
Move2
Trans
Weak
¬Weak
¬Blocked
¬Dead
Dead
¬Trans'd
Trans'd
55

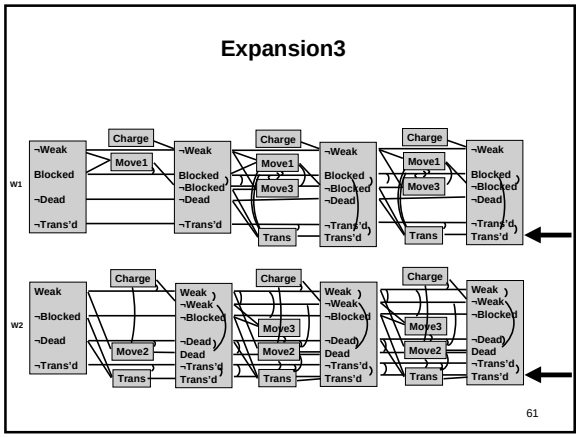

Expansion2
W1
¬Weak
Blocked
¬Dead
¬Trans'd
Charge
Move1
¬Weak
Blocked
¬Blocked
¬Dead
¬Trans'd
Move3
Trans
W2
Weak
¬Blocked
¬Dead
¬Trans'd
Move2
Trans
Weak
¬Weak
¬Blocked
¬Dead
Dead
¬Trans'd
Trans'd

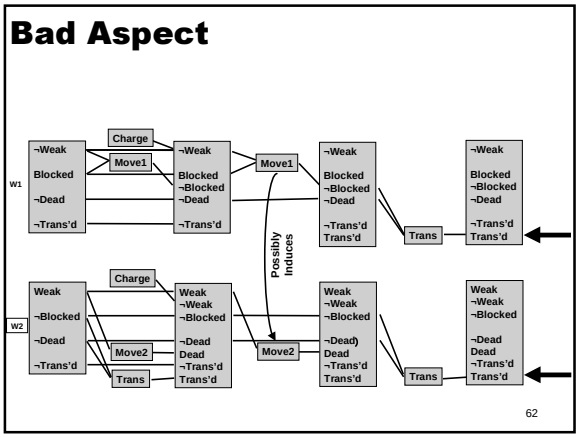

Finding Solutions
W1
¬Weak
Blocked
¬Dead
¬Trans'd
Charge
Move1
Move3
Trans
W2
Weak
¬Blocked
¬Dead
¬Trans'd
Move2
Trans
¬Weak
Blocked
¬Blocked
¬Dead
Dead
¬Trans'd
Trans'd
54

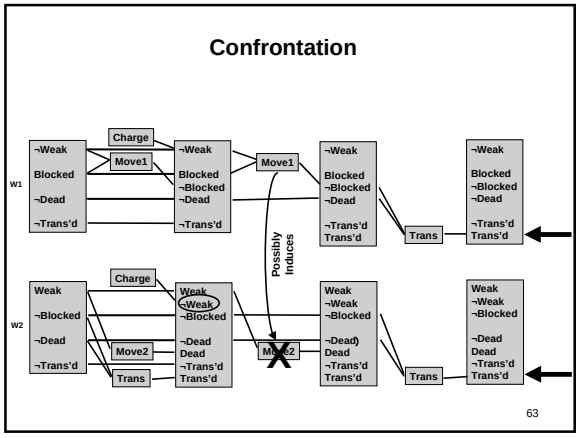

Solution?
W1
¬Weak
Blocked
¬Dead
¬Trans'd
Move1
Trans
W2
Weak
¬Blocked
¬Dead
¬Trans'd
¬Weak
Blocked
¬Blocked
¬Dead
Dead
¬Trans'd
Trans'd
58

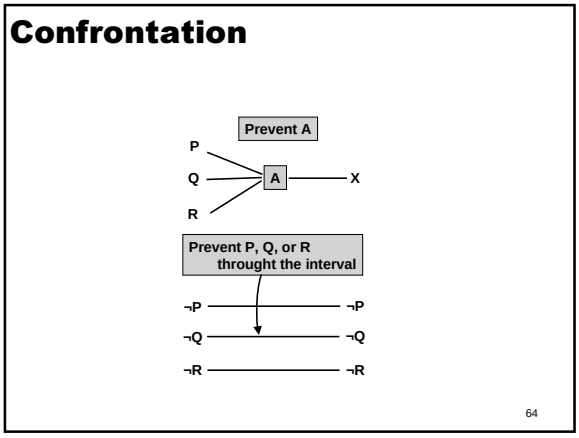

Induced Aspects
W1
¬Weak
Blocked
¬Dead
¬Trans'd
Move1
Induces
Trans
W2
Weak
¬Blocked
¬Dead
¬Trans'd
Move2
Trans
¬Weak
Blocked
¬Blocked
¬Dead
¬Trans'd
Weak
¬Weak
¬Blocked
¬Dead
Dead
¬Trans'd
Trans'd
59

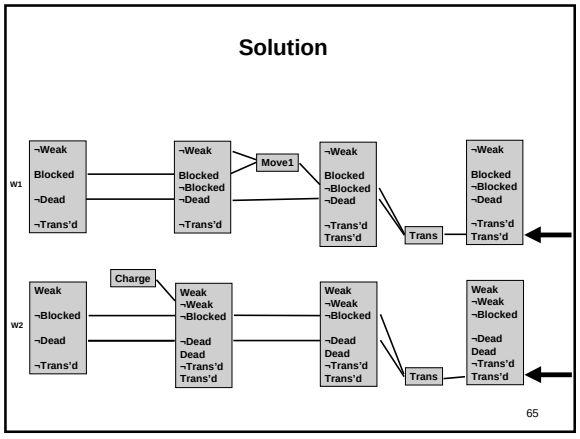

Finding a Solution
If goals are present & non-mutex:
Choose non-mutex action to achieve each goal
Add preconditions to next goal set
Check induced aspects
Not Enough
60

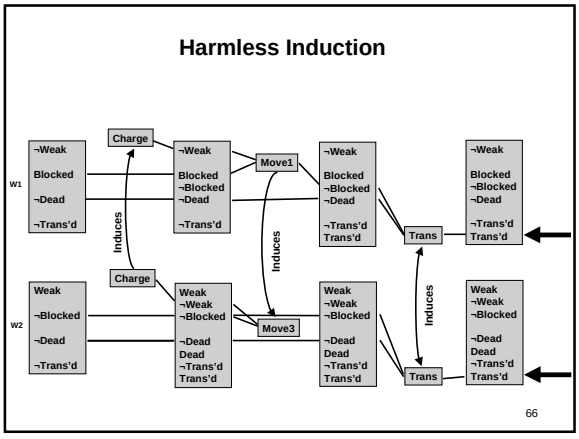

Expansion3
Charge
Move1
Move2
Move3
Trans
¬Weak
Blocked
¬Dead
¬Trans'd
Weak
¬Blocked
Dead
Trans'd
W1
W2
61


Bad Aspect
Charge
Move1
Move2
Trans
Possibly Induces
¬Weak
Blocked
¬Blocked
¬Dead
¬Trans'd
Trans'd
Weak
¬Weak
¬Blocked
Dead
W1
W2
62

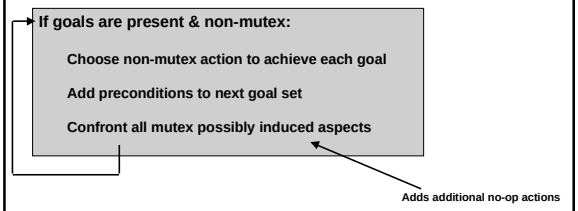

Confrontation
Charge
Move1
Move2
Trans
Possibly Induces
X
¬Weak
Blocked
¬Blocked
¬Dead
¬Trans'd
Trans'd
Weak
¬Weak
¬Blocked
Dead
W1
W2
63


Confrontation
Prevent A
P
Q
R
A
X
Prevent P, Q, or R throught the interval
¬P
¬Q
¬R
64


Solution
Charge
Move1
Trans
¬Weak
Blocked
¬Blocked
¬Dead
¬Trans'd
Trans'd
Weak
¬Weak
¬Blocked
Dead
W1
W2
65

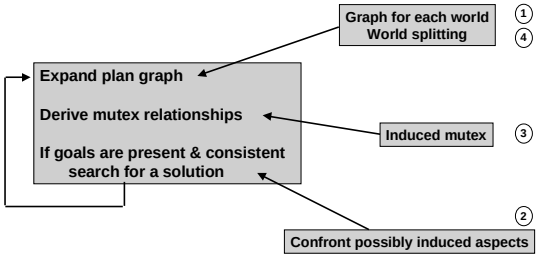

Harmless Induction
Charge
Move1
Move3
Trans
Induces
¬Weak
Blocked
¬Blocked
¬Dead
¬Trans'd
Trans'd
Weak
¬Weak
¬Blocked
Dead
W1
W2
66

Finding a Solution



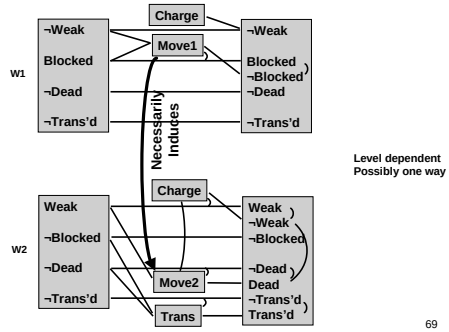
67

Conformant Graphplan



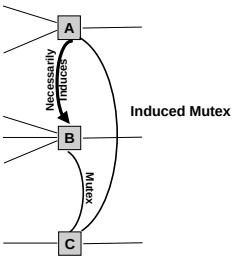
68

Necessarily Induced Aspects



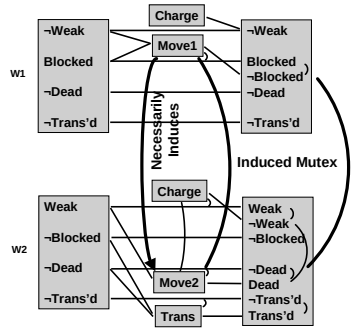
69

Induced Mutex



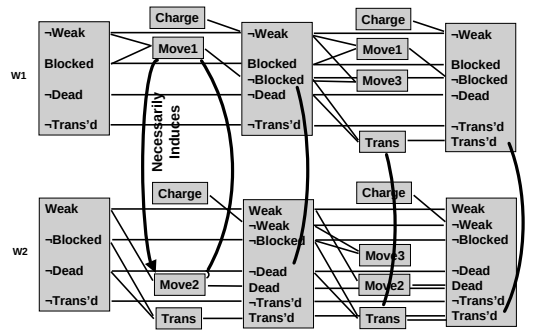
70

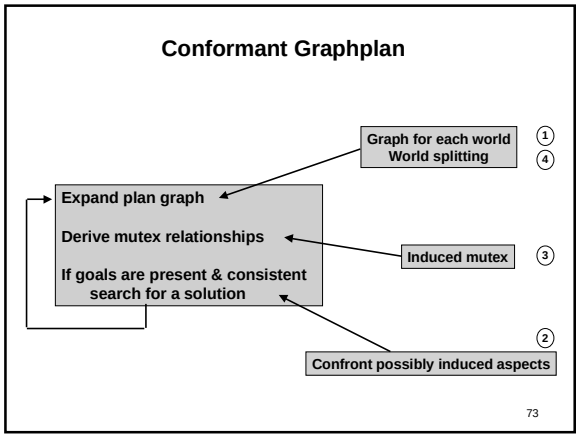
Induced Mutex



71

Induced Mutex




Conformant Graphplan
Graph for each world
World splitting
1
4
Expand plan graph
Derive mutex relationships
If goals are present & consistent
search for a solution
Induced mutex
3
2
Confront possibly induced aspects
73