

Planning, Execution & Learning: Conditional and Universal Planning

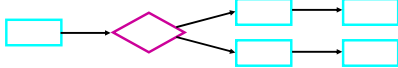
Reid Simmons
Manuela Veloso

Plans

- Classical Plans are *Sequences* or *Partial Orders*
- Conditional Plans are *Branching*
 - Tree structured
- Universal Plans are *Policies*
 - Mappings from states to actions

Conditional Planning

- Create Branching Plans
 - Take *observations* into account when selecting actions
- Observations Used to Handle Uncertainty
 - Uncertainty arises from non-deterministic actions
 - Uncertainty arises from lack of knowledge
- Planners Differ With Respect To:
 - Representation of uncertainty (logic, probabilities)
 - Representation of plans (trees, graphs)
 - Representation of observations
 - Search control



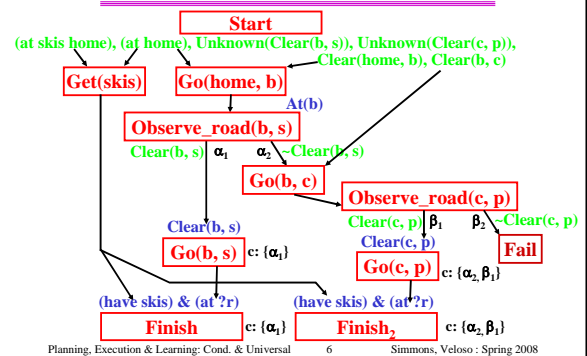
CNLP (Peot & Smith, 1992)

- Extensions to SNLP to Create Conditional Plans with Observations
- Extensions to SNLP Representation
 - Three-valued logic (True, False, *Unknown*)
 - Observation actions
 - Observe_Road(?loc1 ?loc2)
 - Pre: At(?loc1), Unknown(Clear(?loc1, ?loc2))
 - + α_1 : Clear(?loc1, ?loc2)
 - + α_2 : ~Clear(?loc1, ?loc2)
 - Contexts
 - Compatible observation labels

CNLP Extensions to SNLP

- “Conditioning”
 - Can remove threat by *separating contexts* (i.e., making them incompatible)
- Propagation of *context labels* and *reasons*
 - **Contexts**: What actions are incompatible
 - **Reasons**: what goals an action supports
- Tree-structured plan
 - Goal replication

Conditionally Planning a Ski Trip



CNLP Summary

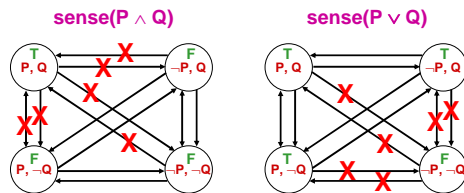
- Can Create Conditional Plans with Observation Actions
 - However, no explicit distinction between observations and causal effects
- Can Handle *Disjunctive Uncertainty*
 - No notion of which conditions more likely
 - Increases search space tremendously
- Can Plan with *Failure* as an Option

Sensory Graphplan (Weld, Anderson & Smith, 1998)

- Adds Sensing Actions to Graphplan
 - Distinguish *sensory* effects from *causal* effects
 - $sense(wff)$ is true if expression wff holds in the possible world
 - For simplicity, *sense* effects cannot occur in actions with preconditions (including being in conditional effects)
- Adds Knowledge Effects
 - $K(\neg u:v)$ is true if, being in world v the agent knows that u is not a consistent (accessible) possible world

Sense Effects

- The Result of a Sense Effect is to Partition the Accessibility Relation Amongst Possible Worlds

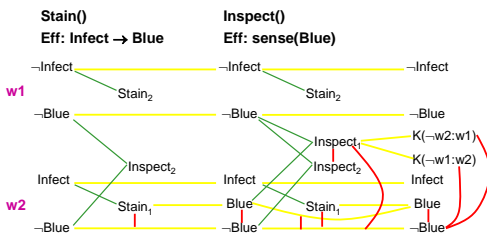


Possible Worlds Plan Graph

- Create Separate Plan Graph for Each Possible World
 - Same algorithm as regular Graphplan
 - One world for each possible initial state
 - Handle non-deterministic effects by splitting possible worlds
 - Can make more efficient by finding mutexes between worlds
- Extract Solution for Goals in Each Possible World
 - Need to consider whether actions chosen for one possible world are mutex in any possible world
 - Can *confront* such interactions by adding negation of preconditions as subgoal

Expanding Plan Graph

- Compute Knowledge Propositions at Each Action Level
 - Link knowledge propositions to sense effects
 - Link sense effects to literals in *all* relevant possible worlds



Solution Extraction

- Add *Conditioning* to CGP Solution Extraction
 - Add as subgoals knowledge propositions that partition aspects that are mutex with previously added aspects
 - Uses constraint satisfaction to maintain contexts
 - Planner does not have to commit to exactly what possible world the system is in
- *The First Conditional Planner with Reasonable Performance!!*

Universal Plans

- **Policy:** State/Action Mapping
 - Specifies (optimal) action to perform in every state
 - Sequences, partial-order plans, conditional plans can all be transformed into *partial policies*
- **Advantages:** Coverage
- **Disadvantages:** Plan size, planning complexity
- How to Represent?
- How to Generate?
- How to Execute?

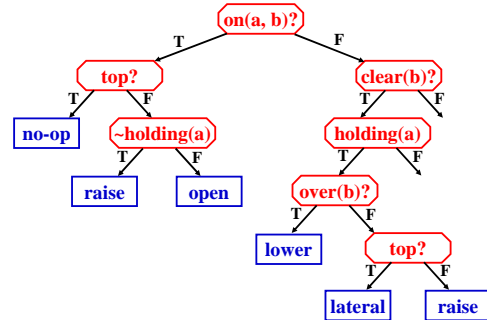
How to Represent Policies

- Table
 - One entry per state
- Decision Tree
 - Branching tree
- Binary Decision Diagram (BDD)
 - Branching DAG

Decision Tree Representation

- Table Representation
 - Need to know values of *all* state variables
 - Access is linear in number of state variables
- Decision Tree Representation
 - Need to know values of only *relevant* state variables
 - Based on knowing values of certain variables, others become irrelevant
 - Access is logarithmic in number of state variables
 - **But**, still need one leaf (action) per state

Universal Block-Stacking Plan



Universal Plans (Schoppers 1987)

- Complete Mapping From Sensors to *Conditions*
 - Can take duration of actions into account
 - Can take advantage of dynamics of environment
 - Influenced by PRS (Georgeoff), REX (Kaelbling), and robotics (control theory)
- Implements Policy as *Decision Tree*
 - Treats planning & plan selection as classification problem
- Can be Synthesized Automatically
 - Uses back-chaining, non-linear planner
 - Break into action-sized chunks, with appropriate sensing actions

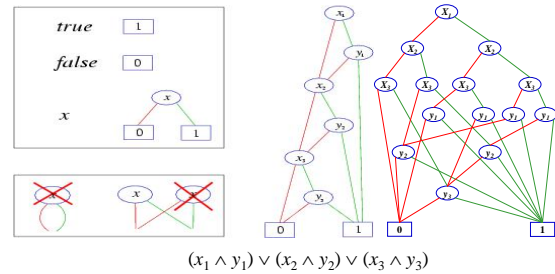
BDD Representation

- Binary-Decision Diagram (Bryant 1985)
 - Directed Acyclic Graph (DAG)
 - Compact representation of Boolean formulae
 - Have been successfully applied in model checking to **implicitly** represent and traverse **very large** state spaces
 - Have recently (since 1998) been shown to be well suited for **universal plans** in non-deterministic domains

BDD Representation

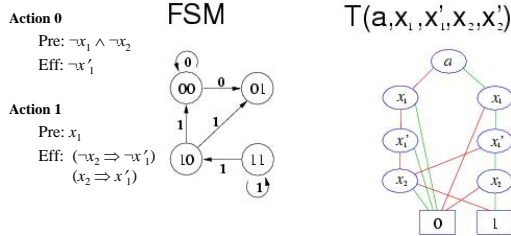
- A BDD is a DAG with:
 - One or two terminal nodes labeled 0 or 1
 - A set of variable nodes with two edges: *low* (false) and *high* (true)
 - A linear ordering of variables.
 - Bad variable ordering leads to exponential size
 - Reductions:
 - Uniqueness of nodes associated to the same variable
 - No redundant tests

BDD Example

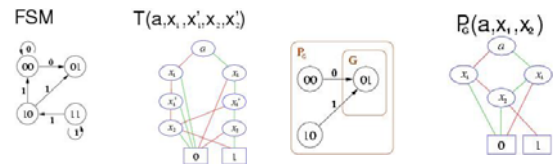


Representing Actions

- Representing the Transition Relation
 - Boolean formula describing relationship between variables in current and next state: $T(a, x, x')$



Pre-Image Computation



$$\text{Pre-Image}(a, x) = \exists x'. T(a, x, x') \wedge V(x')$$

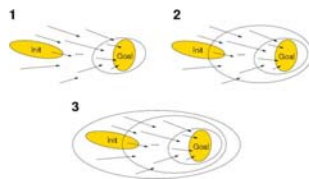
$$P_G(a, x_1, x_2) = \exists x'_1, x'_2. T(a, x_1, x_2, x'_1, x'_2) \wedge (\neg x'_1 \wedge x'_2)$$

$$\text{Post-Image}(a, x) = \exists x. T(a, x, x') \wedge V(x)$$

BDD-Based Planning

- Key Idea: Solve for **Sets** of States

```
function Plan(T, I, G)
  U := 0; V := G
  while I not subset V
    U_c := PreComp(T, V)
    if U_c = 0 then
      return failure
    else
      U := U union U_c
      V := V union states(U)
  return U
```



BDD-Based Planning

- Different “Guarantees” for Generated Policies
 - Depends on strictness of *PreComp* rule

