EECS 496: Sequential Decision Making

Soumya Ray

sray@case.edu

Office: Olin 516

Office hours : T 4-5:30 or by appointment

Recap

- What is the connection between planning and satisfiability?
- What are literals? Clauses? CNFs?
- What is a bounded planning problem?
- We need to create a CNF that represents: (i)______ (ii)______ (iii)_________.
- This CNF must be such that: (i) any truth assignment _______ and (ii) any solution to _____ implies _______.
- The architecture of a SAT planner consists of a _____, a _____, a ______ and a ________.
- How can we detect the absence of a plan in this formulation?
- What is a fluent?
- How do we represent a state in the CNF?
- How do we represent a goal?
- How do we represent actions?

Today

- Planning as Satisfiability
- Nonclassical Planning: Actions with Durations and Resources

State Representation

- A general purpose inference procedure does not have CWA, so we also need to specify all fluents that are *false* at a state

- Final state representation for state S_i:

$$\left(\bigwedge_{f_j \in S_i} f_{ji} \right) \wedge \left(\bigwedge_{f_j \notin S_i} \neg f_{ji} \right)$$

Representing the Goal

- We know that the n^{th} state must satisfy the goal
- Also the goal is just a set of positive literals, so we represent the goal as:

$$\left(\bigwedge_{g \in Goal} g_n \right)$$

Action Representation

- We write a formula that describes what needs to have happened if the i^{th} action in the plan is a_i

$$a_i \Rightarrow \left[\left(\bigwedge_{p_j \in \text{precond}(a)} p_{ji} \right) \wedge \left(\bigwedge_{e_j \in \text{effects}(a)} e_{j,i+1} \right) \right]$$

- Need one such formula for every action *for every step*
- Do we need anything else?

Total Ordering Between Actions

- How do we ensure only one action happens at step i?

- Include *complete exclusion* axioms: for all pairs a_i, b_i

$$\neg\left(a_i \wedge b_i\right) \equiv \neg a_i \vee \neg b_i$$

Frame Axioms

- We also need to specify that the fluents *not affected* by an action *retain their truth value in the next state* (maintenance actions in Graphplan)
  - Otherwise they become “unknown”

- This is the (other part of the) frame problem

Explanatory Frame Axioms

- If a fluent *changes*, one of the actions with that as an effect *must have executed*

$$(\neg f_{ji} \wedge f_{j,i+1}) \Rightarrow \left(\bigvee_{a: f_j \in ADD(a)} a_i \right)$$

$$(f_{ji} \wedge \neg f_{j,i+1}) \Rightarrow \left(\bigvee_{a: f_j \in DEL(a)} a_i \right)$$

Example

- Planning domain:
  - one robot r1
  - two adjacent locations l1, l2
  - one operator (move the robot)
- Encode (P,n) where $n = 1$
  - Initial state: {at(r1,l1)}
    Encoding: at(r1,l1,0) $\wedge$ $\neg$at(r1,l2,0)
  - Goal: {at(r1,l2)}
    Encoding: at(r1,l2,1)

Example (continued)

- Schema: move(r, l, l')

PRE: at(r,l)

ADD: at(r,l')

DEL: at(r,l)

Encoding: (for actions move(r1,l1,l2) and move(r1,l2,l1) at time step 0)

$move(r1,l1,l2,0) \Rightarrow at(r1,l1,0)$

$move(r1,l1,l2,0) \Rightarrow at(r1,l2,1)$

$move(r1,l1,l2,0) \Rightarrow \neg at(r1,l1,1)$

$move(r1,l2,l1,0) \Rightarrow at(r1,l2,0)$

$move(r1,l2,l1,0) \Rightarrow at(r1,l1,1)$

$move(r1,l2,l1,0) \Rightarrow \neg at(r1,l2,1)$

Example (continued)

- Schema: move(r, l, l')

  PRE: at(r,l)

  ADD: at(r,l')

  DEL: at(r,l)

- Complete-exclusion axiom:

  $\neg$move(r1,l1,l2,0) $\vee$ $\neg$move(r1,l2,l1,0)

- Explanatory frame axioms:

  $\neg$at(r1,l1,0) $\wedge$ at(r1,l1,1) $\Rightarrow$ move(r1,l2,l1,0)

  at(r1,l1,0) $\wedge$ $\neg$at(r1,l1,1) $\Rightarrow$ move(r1,l1,l2,0)

  $\neg$at(r1,l2,0) $\wedge$ at(r1,l2,1) $\Rightarrow$ move(r1,l1,l2,0)

  at(r1,l2,0) $\wedge$ $\neg$at(r1,l2,1) $\Rightarrow$ move(r1,l2,l1,0)

Complete Formula for (P,1)

$[\ at\_r1\_l1\_0 \land \neg at\_r1\_l2\_0\ ] \land$
$at\_r1\_l2\_1 \land$
$[\neg\ move\_r1\_l1\_l2\_0 \lor at\_r1\_l1\_0\ ] \land$
$[\neg\ move\_r1\_l1\_l2\_0 \lor at\_r1\_l2\_1\ ] \land$
$[\neg\ move\_r1\_l1\_l2\_0 \lor \neg at\_r1\_l1\_1\ ] \land$
$[\neg\ move\_r1\_l2\_l1\_0 \lor at\_r1\_l2\_0\ ] \land$
$[\neg\ move\_r1\_l2\_l1\_0 \lor at\_r1\_l1\_1\ ] \land$
$[\neg\ move\_r1\_l2\_l1\_0 \lor \neg at\_r1\_l2\_1\ ] \land$
$[\ \neg move\_r1\_l1\_l2\_0 \lor \neg move\_r1\_l2\_l1\_0\ ] \land$
$[at\_r1\_l1\_0 \lor \neg\ at\_r1\_l1\_1 \lor move\_r1\_l2\_l1\_0\ ] \land$
$[at\_r1\_l2\_0 \lor \neg\ at\_r1\_l2\_1 \lor move\_r1\_l1\_l2\_0\ ] \land$
$[\neg\ at\_r1\_l1\_0 \lor at\_r1\_l1\_1 \lor move\_r1\_l1\_l2\_0\ ] \land$
$[\neg\ at\_r1\_l2\_0 \lor at\_r1\_l2\_1 \lor move\_r1\_l2\_l1\_0\ ]$

Input to SAT solver.

Solution for (P,1)

$[\, at\_r1\_l1\_0 \wedge \neg at\_r1\_l2\_0 \,] \wedge$
$at\_r1\_l2\_1 \wedge$
$[\neg\, move\_r1\_l1\_l2\_0 \vee at\_r1\_l1\_0 \,] \wedge$
$[\neg\, move\_r1\_l1\_l2\_0 \vee at\_r1\_l2\_1 \,] \wedge$
$[\neg\, move\_r1\_l1\_l2\_0 \vee \neg at\_r1\_l1\_1 \,] \wedge$
$[\neg\, move\_r1\_l2\_l1\_0 \vee at\_r1\_l2\_0 \,] \wedge$
$[\neg\, move\_r1\_l2\_l1\_0 \vee at\_r1\_l1\_1 \,] \wedge$
$[\neg\, move\_r1\_l2\_l1\_0 \vee \neg at\_r1\_l2\_1 \,] \wedge$
$[\, \neg move\_r1\_l1\_l2\_0 \vee \neg move\_r1\_l2\_l1\_0 \,] \wedge$
$[at\_r1\_l1\_0 \vee \neg\, at\_r1\_l1\_1 \vee move\_r1\_l2\_l1\_0 \,] \wedge$
$[at\_r1\_l2\_0 \vee \neg\, at\_r1\_l2\_1 \vee move\_r1\_l1\_l2\_0 \,] \wedge$
$[\neg\, at\_r1\_l1\_0 \vee at\_r1\_l1\_1 \vee move\_r1\_l1\_l2\_0 \,] \wedge$
$[\neg\, at\_r1\_l2\_0 \vee at\_r1\_l2\_1 \vee move\_r1\_l2\_l1\_0 \,]$

Extracting a Plan

- Suppose we find an assignment of truth values that satisfies the formula
  - This means *P* has a solution of length *n*
- For $i=0,\ldots,n-1$, there will be exactly one action *a* such that $a_i = true$
  - This is the i^{th} action of the plan
- Example (from the previous slides):
  - Can be satisfied with move(r1,l1,l2,0) = *true*
  - Thus ⟨move(r1,l1,l2,0)⟩ is a solution for (*P*,0)
    - It's the only solution - no other way to satisfy

Supporting Layered Plans

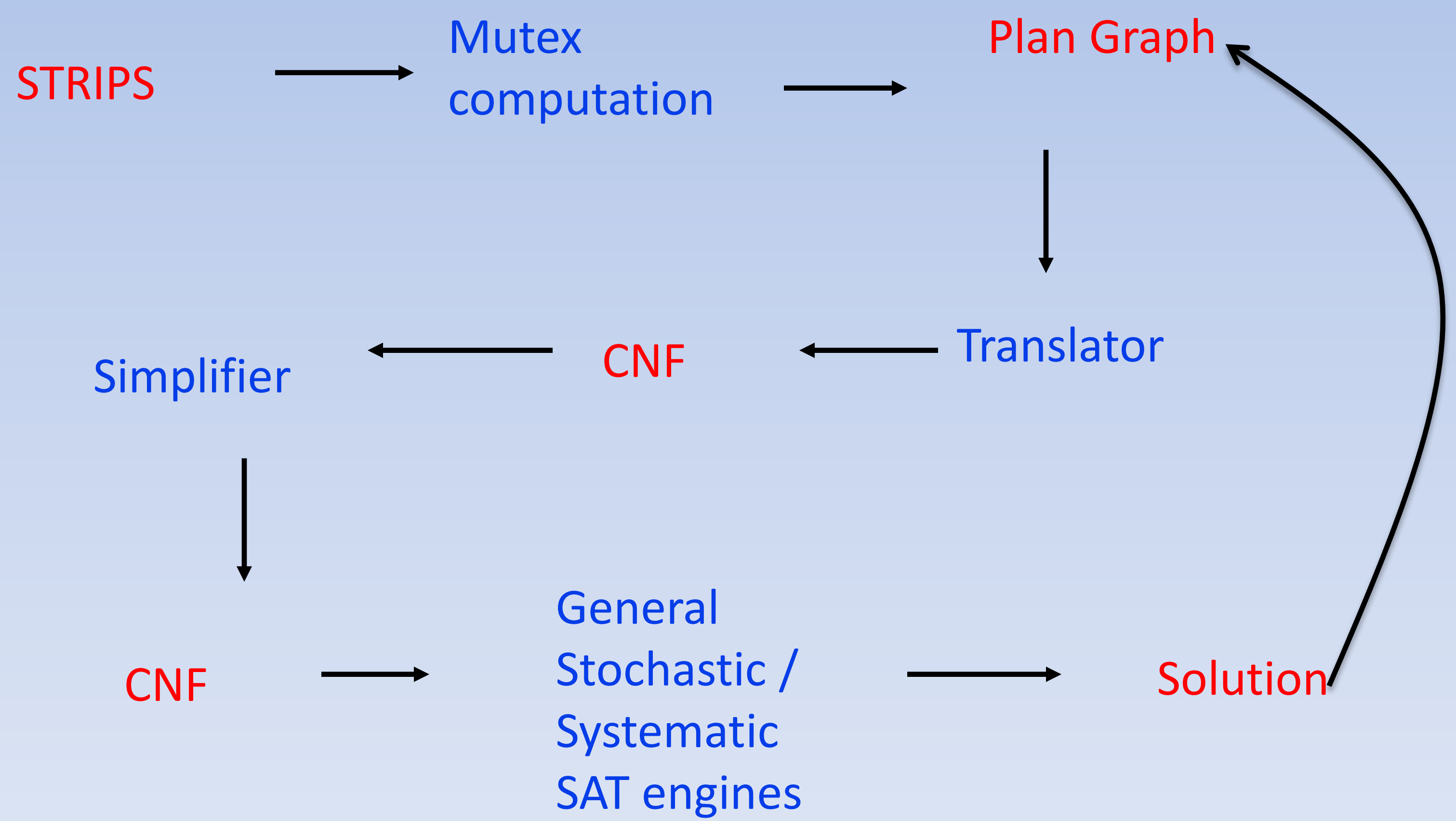
- *Complete exclusion* axiom:
  - For **<u>all</u>** actions *a* and *b* and time steps *i* include the formula $\neg a_i \vee \neg b_i$
  - this guaranteed that there could be only one action at a time
- *Partial exclusion axiom*:
  - For any pair of *mutex* actions *a* and *b* and each time step *i* include the formula $\neg a_i \vee \neg b_i$
  - This encoding will allow for more than one action to be taken at a time step resulting in layered plans
  - This is advantageous because fewer time steps are required (i.e. shorter formulas)

BlackBox (GraphPlan + SATPlan)

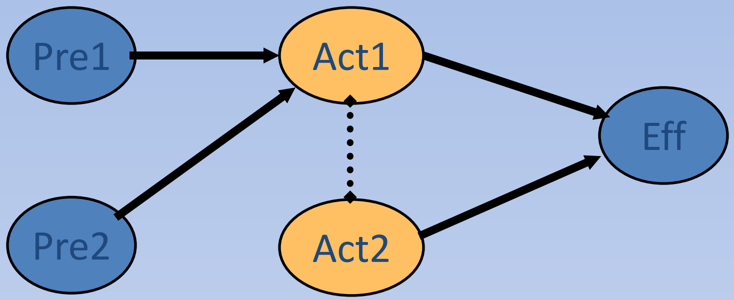
- The BlackBox procedure combines planning-graph expansion and satisfiability checking
- For layer n = 0, 1, 2, …
  - *Graph expansion:*
    - create a planning graph that contains n layers
  - Check whether the planning graph satisfies the condition for plan existence
  - If it does, then
    - Encode (P,n) as a satisfiability problem Φ but include only the actions in the planning graph
    - If Φ is satisfiable then return the solution

Blackbox

Can be thought of as an implementation of GraphPlan that uses an alternative plan extraction technique than the backward chaining of GraphPlan.



Translation of Planning Graph



$$\text{Eff} \Rightarrow \text{Act1} \vee \text{Act2}$$

$$\text{Act1} \Rightarrow \text{Pre1} \wedge \text{Pre2}$$

$$\neg\text{Act1} \vee \neg\text{Act2}$$

Can create such constraints for every node in the planning graph

What SATPLAN Shows

- General propositional reasoning can compete with state of the art specialized planning systems
  - Radically new stochastic approaches to SAT can provide very low exponential scaling
- Why does it work well?
  - More flexible than forward or backward state space planning
  - Randomized algorithms less likely to get trapped along bad paths