Lecture 6b: Search Based Planning

CSCI 360

Introduction to Artificial Intelligence

USC

Here is where we are…

Week	30000D	30282R	Topics	Chapters
1	1/7	1/8	Intelligent Agents	[Ch 1.1-1.4 and 2.1-2.4]
	1/9	1/10	Problem Solving and Search	[Ch 3.1-3.3]
2	1/14	1/15	Uninformed Search	[Ch 3.3-3.4]
	1/16	1/17	Heuristic Search (A*)	[Ch 3.5]
3	1/21	1/22	Heuristic Functions	[Ch 3.6]
	1/23	1/24	Local Search	[Ch 4.1-4.2]
	1/25		Project 1 Out	
4	1/28	1/29	Adversarial Search	[Ch 5.1-5.3]
	1/30	1/31	Knowledge Based Agents	[Ch 7.1-7.3]
5	2/4	2/5	Propositional Logic Inference	[Ch 7.4-7.5]
	2/6	2/7	First-Order Logic	[Ch 8.1-8.4]
	2/8		Project 1 Due	
	2/8		Homework 1 Out	
6	2/11	2/12	Rule-Based Systems	[Ch 9.3-9.4]
	2/13	2/14	Search-Based Planning	[Ch 10.1-10.3]
	2/15		Homework 1 Due	
7	2/18	2/19	SAT-Based Planning	[Ch 10.4]
	2/20	2/21	Knowledge Representation	[Ch 12.1-12.5]
8	2/25	2/26	Midterm Review	
	2/27	2/28	**Midterm Exam**	

Outline

- What is AI?
- Problem-solving agent
  - Uninformed (DFS), informed (A*), and local search
  - Adversarial search (minimax, alpha-beta pruning)
- **Knowledge-based agent**
  - Propositional Logic
  - First Order Logic (FOL)
  - **Automated Reasoning in FOL**
    - Substitution
    - Unification (GMP)
    - Chaining (forward and backward)
    - Resolution

Resolution (a simple example)

KB:

(1) father (art, jon)

(2) father (bob, kim)

(3) father (X, Y) $\Rightarrow$ parent (X, Y)

Goal: parent (art, jon)?

(KB) $\wedge$ ($\neg$ Goal) is "Unsatisfiable"

Resolution (a simple example)

KB:

(1) father (art, jon)
(2) father (bob, kim)
(3) father (X, Y) ⇒ parent (X, Y)

Goal: parent (art, jon)?

¬ parent(art, jon)   father(X, Y) => parent(X, Y)
\   /
¬ parent(art, jon)   ¬ father(X, Y) ∨ parent(X, Y)
\   /
¬ father (art, jon)   father (art, jon)
\   /
[ ]

FOL resolution rule

$$\textsc{Unify}(\ell_i, \neg m_j) = \theta.$$

$$\boxed{\frac{\ell_1 \vee \cdots \vee \ell_k, \qquad m_1 \vee \cdots \vee m_n}{\textsc{Subst}(\theta, \ell_1 \vee \cdots \vee \ell_{i-1} \vee \ell_{i+1} \vee \cdots \vee \ell_k \vee m_1 \vee \cdots \vee m_{j-1} \vee m_{j+1} \vee \cdots \vee m_n)}}$$

- Example:

$$[Animal(F(x)) \vee Loves(G(x), x)] \quad \text{and} \quad [\neg Loves(u, v) \vee \neg Kills(u, v)]$$

$$\Big\downarrow \; \theta = \{u/G(x), v/x\}$$

$$[Animal(F(x)) \vee \neg Kills(G(x), x)]$$

FOL resolution (example)

$$\frac{\begin{array}{l}\neg Rich(x) \vee Unhappy(x) \\ Rich(Me)\end{array}}{Unhappy(Me)}$$

with $\theta = \{x/Me\}$

FOL Conjunctive normal form (CNF)

Steps:

1.
2.
3.
4.
5.
6.
7.

FOL Conjunctive normal form (CNF)

Steps:

1. Replace $P \Rightarrow Q$ by $\neg P \vee Q$
2. Move $\neg$ inwards, e.g., $\neg \forall x\, P$ becomes $\exists x\, \neg P$
3. Standardize variables apart, e.g., $\forall x\, P \vee \exists x\, Q$ becomes $\forall x\, P \vee \exists y\, Q$
4. Move quantifiers left in order, e.g., $\forall x\, P \vee \exists x\, Q$ becomes $\forall x \exists y\, P \vee Q$
5. Eliminate $\exists$ by Skolemization (next slide)
6. Drop universal quantifiers
7. Distribute $\wedge$ over $\vee$, e.g., $(P \wedge Q) \vee R$ becomes $(P \vee Q) \wedge (P \vee R)$

Skolemization

- Why can't (y) be replaced by a constant symbol?

$$\forall x \;\; [\exists y \;\; Animal(y) \wedge \neg Loves(x,y)] \vee [\exists z \;\; Loves(z,x)]$$

  - Everyone loves the same animal (F), and the same (G) loves everyone

$$\boxed{\forall x \;\; [Animal(F) \wedge \neg Loves(x,F)] \vee Loves(G, x)}$$

- Each person loves a different animal *F(x)*, and a different *G(x)* loves each person

$$\boxed{\forall x \;\; [Animal(F(x)) \wedge \neg Loves(x,F(x))] \vee Loves(G(x),x)}$$

- Distribution

$$\forall x \;\; [Animal(F(x)) \vee Loves(G(x),x) \;] \wedge [\neg Loves(x,F(x)) \vee Loves(G(x),x) \;]$$

Converting to CNF

$$\boxed{\forall x \ [\forall y \ \ Animal(y) \ \Rightarrow \ Loves(x,y)] \ \Rightarrow \ [\exists y \ \ Loves(y,x)]}$$

- Eliminate implications

$$\forall x \ [\neg\forall y \ \ \neg Animal(y) \lor Loves(x,y)] \lor [\exists y \ \ Loves(y,x)]$$

- Move negation inwards

$$\forall x \ [\exists y \ \ \neg(\neg Animal(y) \lor Loves(x,y))] \lor [\exists y \ \ Loves(y,x)] \ .$$
$$\forall x \ [\exists y \ \ \neg\neg Animal(y) \land \neg Loves(x,y)] \lor [\exists y \ \ Loves(y,x)] \ .$$
$$\forall x \ [\exists y \ \ Animal(y) \land \neg Loves(x,y)] \lor [\exists y \ \ Loves(y,x)] \ .$$

- Standardize variables

$$\forall x \ [\exists y \ \ Animal(y) \land \neg Loves(x,y)] \lor [\exists z \ \ Loves(z,x)]$$

- Skolemization

$$\forall x \ [Animal(F(x)) \land \neg Loves(x,F(x))] \lor Loves(G(x),x)$$

FOL resolution (another example)

A. $\boxed{\forall x\ \ [\forall y\ \ Animal(y) \Rightarrow Loves(x,y)] \Rightarrow [\exists y\ \ Loves(y,x)]}$

Transform to CNF

FOL resolution (another example)

A. $\forall x \ [\forall y \ \ Animal(y) \Rightarrow Loves(x,y)] \Rightarrow [\exists y \ \ Loves(y,x)]$

Transform to CNF

$$\exists y \neg(Animal(y) \rightarrow Loves(x,y)) \vee (\exists y \ Loves(y,x))$$

$$\exists y \neg (Animal(y) \vee Loves(x,y)) \vee (\exists y \ Loves(y,x))$$

$$\neg (Animal(F(x)) \vee Loves(x,F(x))) \vee (Loves(G(x),x))$$

$$(\neg Animal(F(x)) \wedge \neg Loves(x,F(x))) \vee (Loves(G(x),x))$$

$$(\neg Animal(F(x)) \vee Loves(G(x),x)) \wedge (\neg Loves(x,F(x)) \vee Loves(G(x),x))$$

FOL resolution (another example)

A. $\forall x \;\; [\forall y \;\; Animal(y) \;\Rightarrow\; Loves(x, y)] \;\Rightarrow\; [\exists y \;\; Loves(y, x)]$

B. $\boxed{\forall x \;\; [\exists z \;\; Animal(z) \wedge Kills(x, z)] \;\Rightarrow\; [\forall y \;\; \neg Loves(y, x)]}$

C. $\forall x \;\; Animal(x) \;\Rightarrow\; Loves(Jack, x)$

Transform to CNF

$$\neg\, (\exists z \;\; Animal(z) \wedge Kills(x, z)\,) \vee (\,\forall y \neg\, Loves\,(y, x)\,)$$

$$(\forall z \neg\, Animal(z) \vee \neg\, Kills(x, z)\,) \vee (\,\forall y \neg\, Loves\,(y, x)\,)$$

$$\forall y\, \forall z\, (\,\neg\, Animal(z) \vee \neg\, Kills(x, z) \;\vee\; \neg\, Loves\,(y, x)\,)$$

FOL resolution (another example)

A. $\forall x \; [\forall y \;\; Animal(y) \; \Rightarrow \; Loves(x,y)] \; \Rightarrow \; [\exists y \;\; Loves(y,x)]$

B. $\forall x \; [\exists z \;\; Animal(z) \wedge Kills(x,z)] \; \Rightarrow \; [\forall y \;\; \neg Loves(y,x)]$

C. $\forall x \;\; Animal(x) \; \Rightarrow \; Loves(Jack, x)$

D. $Kills(Jack, Tuna) \vee Kills(Curiosity, Tuna)$

E. $Cat(Tuna)$

F. $\forall x \;\; Cat(x) \Rightarrow Animal(x)$

¬G. $\neg Kills(Curiosity, Tuna)$

A1. $Animal(F(x)) \vee Loves(G(x), x)$

A2. $\neg Loves(x, F(x)) \vee Loves(G(x), x)$

B. $\neg Loves(y, x) \vee \neg Animal(z) \vee \neg Kills(x, z)$

C. $\neg Animal(x) \vee Loves(Jack, x)$

D. $Kills(Jack, Tuna) \vee Kills(Curiosity, Tuna)$

E. $Cat(Tuna)$

F. $\neg Cat(x) \vee Animal(x)$

¬G. $\neg Kills(Curiosity, Tuna)$

Outline

- What is AI?
- Problem-solving agent
  - Uninformed (DFS), informed (A*), and local search
  - Adversarial search (minimax, alpha-beta pruning)
- **Knowledge-based agent**
  - Propositional Logic
  - First Order Logic (FOL)
  - **Search Based Planning**

What we have so far

- Can **TELL** (KB) about new percepts about the world
- (KB) maintains model of the current world state
- Can **ASK** (KB) about any fact that can be inferred from KB

How to use these components to build a ***planning agent***?

i.e., an agent that constructs a plan to achieve a goal

Example: Robot Manipulators

– Example: (courtesy of Martin Rohrmeier)

Difference: “search” vs “planning”

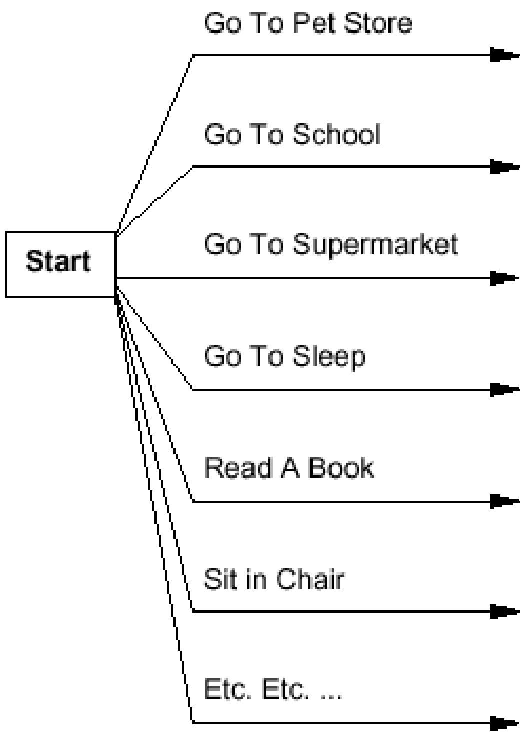
- **Problem-solving agent** can find a sequence of actions that result in a goal state
  - it deals with “**atomic”** representations of states
  - Needs **“domain-specific”** heuristics to perform well in search

- **Planning agent** can also find a sequence of actions that result in a goal state
  - But it uses a **“factored”** representations of states
  - Can have **“generic”** heuristics for search

Logic formulas in a restricted format

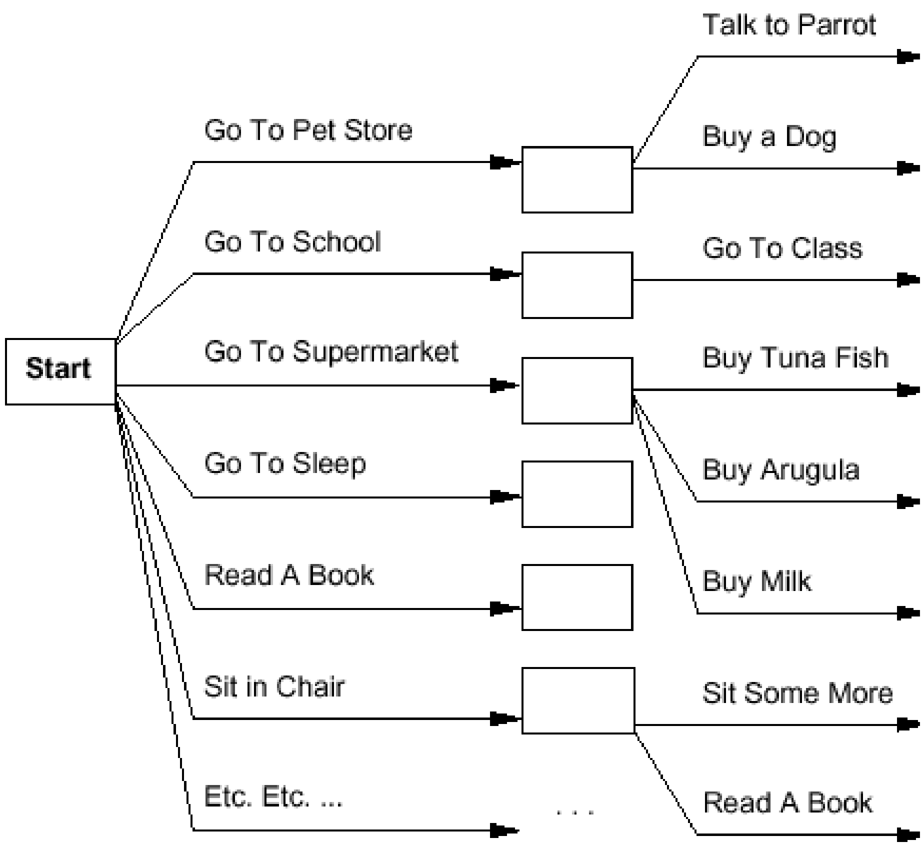
Search vs. planning (example)

- Consider the task **buy milk, bananas, and a cordless drill,** existing search algorithms may fail miserably…



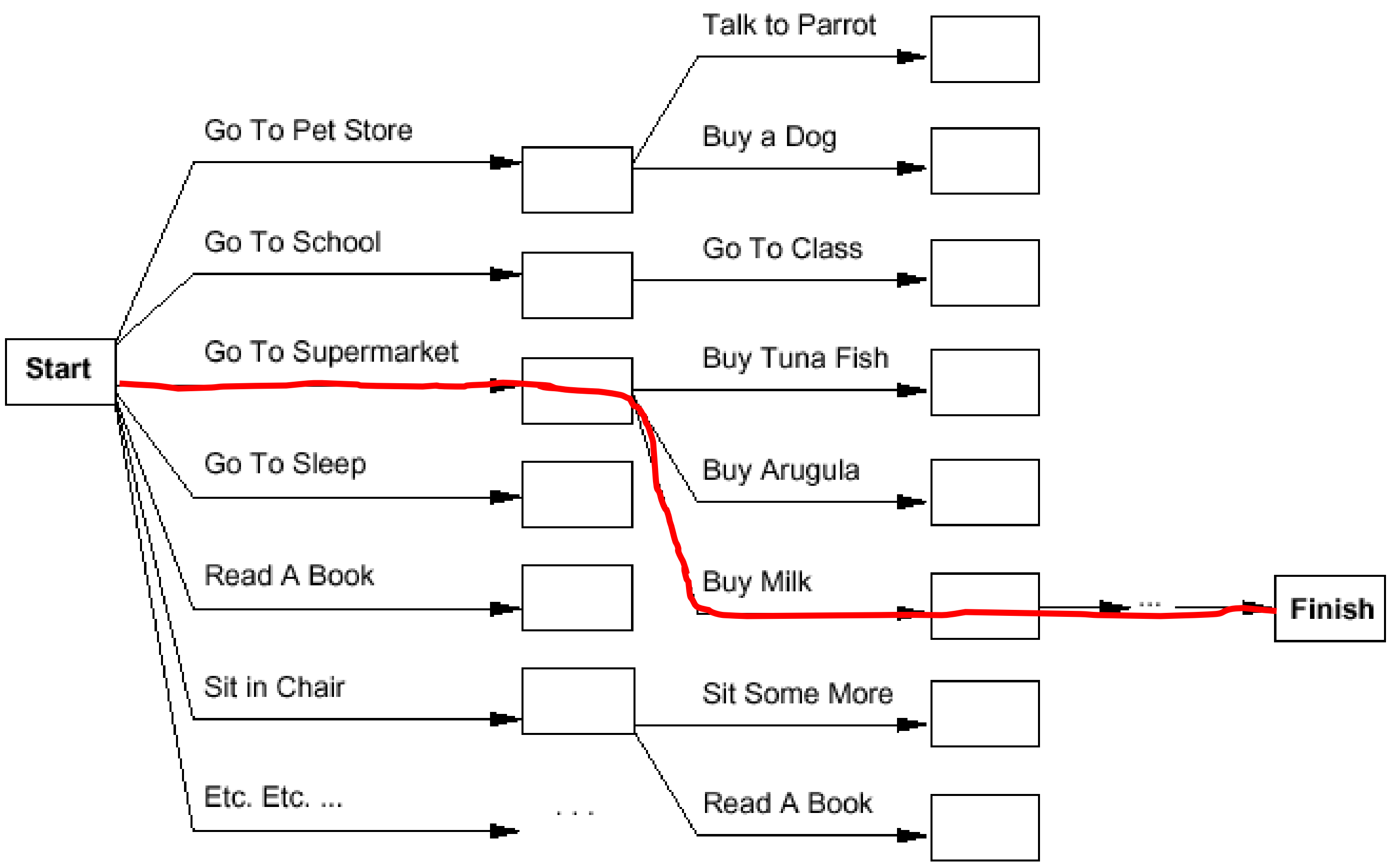
Search vs. planning (example)

- Consider the task **buy milk, bananas, and a cordless drill,** existing search algorithms may fail miserably…



Search vs. planning (example)

- Consider the task **buy milk, bananas, and a cordless drill,** existing search algorithms may fail miserably…



Search vs. planning

- Planning opens up **action** and **goal** representations

	Search	Planning
States		
Actions		
Goal		
Plan		

Search vs. planning

- Planning opens up **action** and **goal** representations

	Search	**Planning**
States	data structures	Logical sentences
Actions	code	Preconditions/outcomes
Goal	code	Logical sentence (conjunction)
Plan	Sequence from S_0	Constraints on actions

PDDL: *Planning Domain Definition Language*

- It uses a restricted subset of first-order logic (FOL) to make planning **efficiently solvable**

- **State:** a conjunction of functionless ground literals

$$Poor \wedge Unknown$$

$$At(Truck_1, Melbourne) \wedge At(Truck_2, Sydney)$$

$At(x, y)$ cannot have variables (x,y)

$At(Father(Fred), Sydney)$ cannot have function symbol

- **Goal:** a conjunction of literals, but may have variables

$$At(Home) \wedge Have(Milk) \wedge Have(Bananas) \wedge Have(Drill)$$

$$At(x) \wedge Sells(x, Milk)$$

PDDL: *Planning Domain Definition Language*

- It uses a restricted subset of first-order logic (FOL) to make planning **efficiently solvable**

- **State:** a conjunction of functionless ground literals

- **Actions:**

Action name

$$
\begin{aligned}
&Action(Fly(p, from, to),\\
&\quad \text{PRECOND:} At(p, from) \wedge Plane(p) \wedge Airport(from) \wedge Airport(to)\\
&\quad \text{EFFECT:} \neg At(p, from) \wedge At(p, to))
\end{aligned}
$$

Conjunction of ***positive*** literals

Conjunction of literals ***(positive or negative)***

PDDL: *Planning Domain Definition Language*

- It uses a restricted subset of first-order logic (FOL) to make planning **efficiently solvable**

- **State:** a conjunction of functionless ground literals

- **Actions:**

$$
\begin{aligned}
&Action(Fly(p, from, to),\\
&\quad \text{PRECOND:} At(p, from) \land Plane(p) \land Airport(from) \land Airport(to)\\
&\quad \text{EFFECT:} \neg At(p, from) \land At(p, to))
\end{aligned}
$$

Negative literal

DEL this lieteral
from the new state

Negative literal

ADD this lieteral
into the new state

PDDL: *Planning Domain Definition Language*

- It uses a restricted subset of first-order logic (FOL) to make planning **efficiently solvable**

- **State:** a conjunction of functionless ground literals

- **Actions:**

$$\boxed{\begin{array}{l} Action(Fly(p, from, to), \\ \quad \text{PRECOND:} At(p, from) \wedge Plane(p) \wedge Airport(from) \wedge Airport(to) \\ \quad \text{EFFECT:} \neg At(p, from) \wedge At(p, to)) \end{array}}$$

- **Transition model:**

$$\text{RESULT}(s, a) = (s - \text{DEL}(a)) \cup \text{ADD}(a)$$

Example: *Air cargo transportation planning*

Example: *Air cargo transportation planning*

$Init(At(C_1, SFO) \wedge At(C_2, JFK) \wedge At(P_1, SFO) \wedge At(P_2, JFK)$
$\wedge\ Cargo(C_1) \wedge Cargo(C_2) \wedge Plane(P_1) \wedge Plane(P_2)$
$\wedge\ Airport(JFK) \wedge Airport(SFO))$

$Goal(At(C_1, JFK) \wedge At(C_2, SFO))$

$Action(Load(c, p, a),$
PRECOND: $At(c, a) \wedge At(p, a) \wedge Cargo(c) \wedge Plane(p) \wedge Airport(a)$
EFFECT: $\neg At(c, a) \wedge In(c, p))$

$Action(Unload(c, p, a),$
PRECOND: $In(c, p) \wedge At(p, a) \wedge Cargo(c) \wedge Plane(p) \wedge Airport(a)$
EFFECT: $At(c, a) \wedge \neg In(c, p))$

$Action(Fly(p, from, to),$
PRECOND: $At(p, from) \wedge Plane(p) \wedge Airport(from) \wedge Airport(to)$
EFFECT: $\neg At(p, from) \wedge At(p, to))$

Example: *Air cargo transportation planning*

$$
\begin{aligned}
&\mathit{Init}(\mathit{At}(C_1, \mathit{SFO}) \wedge \mathit{At}(C_2, \mathit{JFK}) \wedge \mathit{At}(P_1, \mathit{SFO}) \wedge \mathit{At}(P_2, \mathit{JFK}) \\
&\quad \wedge \mathit{Cargo}(C_1) \wedge \mathit{Cargo}(C_2) \wedge \mathit{Plane}(P_1) \wedge \mathit{Plane}(P_2) \\
&\quad \wedge \mathit{Airport}(\mathit{JFK}) \wedge \mathit{Airport}(\mathit{SFO})) \\
&\boxed{\mathit{Goal}(\mathit{At}(C_1, \mathit{JFK}) \wedge \mathit{At}(C_2, \mathit{SFO}))} \\
&\mathit{Action}(\mathit{Load}(c, p, a), \\
&\quad \text{PRECOND: } \mathit{At}(c, a) \wedge \mathit{At}(p, a) \wedge \mathit{Cargo}(c) \wedge \mathit{Plane}(p) \wedge \mathit{Airport}(a) \\
&\quad \text{EFFECT: } \neg \mathit{At}(c, a) \wedge \mathit{In}(c, p)) \\
&\mathit{Action}(\mathit{Unload}(c, p, a), \\
&\quad \text{PRECOND: } \mathit{In}(c, p) \wedge \mathit{At}(p, a) \wedge \mathit{Cargo}(c) \wedge \mathit{Plane}(p) \wedge \mathit{Airport}(a) \\
&\quad \text{EFFECT: } \mathit{At}(c, a) \wedge \neg \mathit{In}(c, p)) \\
&\mathit{Action}(\mathit{Fly}(p, \mathit{from}, \mathit{to}), \\
&\quad \text{PRECOND: } \mathit{At}(p, \mathit{from}) \wedge \mathit{Plane}(p) \wedge \mathit{Airport}(\mathit{from}) \wedge \mathit{Airport}(\mathit{to}) \\
&\quad \text{EFFECT: } \neg \mathit{At}(p, \mathit{from}) \wedge \mathit{At}(p, \mathit{to}))
\end{aligned}
$$

Example: *Air cargo transportation planning*

$$
\begin{aligned}
& Init(At(C_1, SFO) \wedge At(C_2, JFK) \wedge At(P_1, SFO) \wedge At(P_2, JFK) \\
& \quad \wedge Cargo(C_1) \wedge Cargo(C_2) \wedge Plane(P_1) \wedge Plane(P_2) \\
& \quad \wedge Airport(JFK) \wedge Airport(SFO)) \\
& Goal(At(C_1, JFK) \wedge At(C_2, SFO)) \\
& Action(Load(c, p, a), \\
& \quad \text{PRECOND: } At(c, a) \wedge At(p, a) \wedge Cargo(c) \wedge Plane(p) \wedge Airport(a) \\
& \quad \text{EFFECT: } \neg At(c, a) \wedge In(c, p)) \\
& Action(Unload(c, p, a), \\
& \quad \text{PRECOND: } In(c, p) \wedge At(p, a) \wedge Cargo(c) \wedge Plane(p) \wedge Airport(a) \\
& \quad \text{EFFECT: } At(c, a) \wedge \neg In(c, p)) \\
& Action(Fly(p, from, to), \\
& \quad \text{PRECOND: } At(p, from) \wedge Plane(p) \wedge Airport(from) \wedge Airport(to) \\
& \quad \text{EFFECT: } \neg At(p, from) \wedge At(p, to))
\end{aligned}
$$

Example: *Air cargo transportation planning*

$Init(At(C_1, SFO) \wedge At(C_2, JFK) \wedge At(P_1, SFO) \wedge At(P_2, JFK)$
$\quad \wedge Cargo(C_1) \wedge Cargo(C_2) \wedge Plane(P_1) \wedge Plane(P_2)$
$\quad \wedge Airport(JFK) \wedge Airport(SFO))$
$Goal(At(C_1, JFK) \wedge At(C_2, SFO))$
$Action(Load(c, p, a),$
  PRECOND: $At(c, a) \wedge At(p, a) \wedge Cargo(c) \wedge Plane(p) \wedge Airport(a)$
  EFFECT: $\neg At(c, a) \wedge In(c, p))$
$Action(Unload(c, p, a),$
  PRECOND: $In(c, p) \wedge At(p, a) \wedge Cargo(c) \wedge Plane(p) \wedge Airport(a)$
  EFFECT: $At(c, a) \wedge \neg In(c, p))$
$Action(Fly(p, from, to),$
  PRECOND: $At(p, from) \wedge Plane(p) \wedge Airport(from) \wedge Airport(to)$
  EFFECT: $\neg At(p, from) \wedge At(p, to))$

Example: *Air cargo transportation planning*

$Init(At(C_1, SFO) \land At(C_2, JFK) \land At(P_1, SFO) \land At(P_2, JFK)$
$\land\ Cargo(C_1) \land Cargo(C_2) \land Plane(P_1) \land Plane(P_2)$
$\land\ Airport(JFK) \land Airport(SFO))$

$Goal(At(C_1, JFK) \land At(C_2, SFO))$

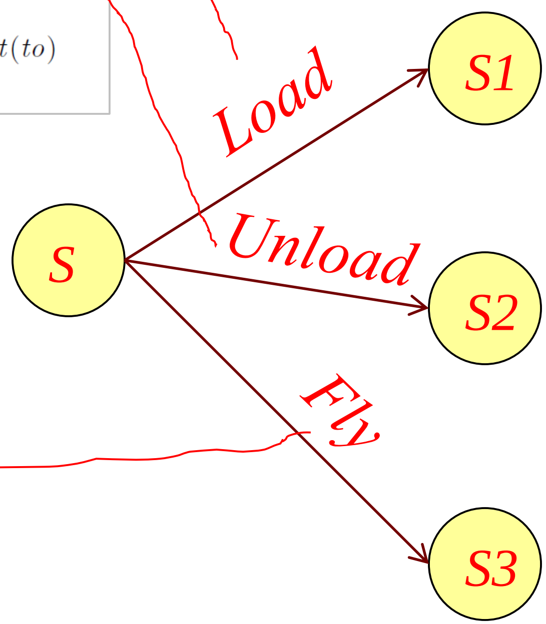
$Action(Load(c, p, a),$
PRECOND: $At(c, a) \land At(p, a) \land Cargo(c) \land Plane(p) \land Airport(a)$
EFFECT: $\neg\, At(c, a) \land In(c, p))$

$Action(Unload(c, p, a),$
PRECOND: $In(c, p) \land At(p, a) \land Cargo(c) \land Plane(p) \land Airport(a)$
EFFECT: $At(c, a) \land \neg\, In(c, p))$

$Action(Fly(p, from, to),$
PRECOND: $At(p, from) \land Plane(p) \land Airport(from) \land Airport(to)$
EFFECT: $\neg\, At(p, from) \land At(p, to))$

Example: *Air cargo transportation planning*

$Init(At(C_1, SFO) \wedge At(C_2, JFK) \wedge At(P_1, SFO) \wedge At(P_2, JFK)$
$\quad \wedge\ Cargo(C_1) \wedge Cargo(C_2) \wedge Plane(P_1) \wedge Plane(P_2)$
$\quad \wedge\ Airport(JFK) \wedge Airport(SFO))$
$Goal(At(C_1, JFK) \wedge At(C_2, SFO))$
$Action(Load(c, p, a),$
PRECOND: $At(c, a) \wedge At(p, a) \wedge Cargo(c) \wedge Plane(p) \wedge Airport(a)$
EFFECT: $\neg At(c, a) \wedge In(c, p))$
$Action(Unload(c, p, a),$
PRECOND: $In(c, p) \wedge At(p, a) \wedge Cargo(c) \wedge Plane(p) \wedge Airport(a)$
EFFECT: $At(c, a) \wedge \neg In(c, p))$
$Action(Fly(p, from, to),$
PRECOND: $At(p, from) \wedge Plane(p) \wedge Airport(from) \wedge Airport(to)$
EFFECT: $\neg At(p, from) \wedge At(p, to))$



The following plan is a solution to the problem:

$$[Load(C_1, P_1, SFO), Fly(P_1, SFO, JFK), Unload(C_1, P_1, JFK),$$
$$Load(C_2, P_2, JFK), Fly(P_2, JFK, SFO), Unload(C_2, P_2, SFO)]\ .$$

Example: *Changing the spare tire*

Example: *Changing the spare tire*

$Init(Tire(Flat) \land Tire(Spare) \land At(Flat, Axle) \land At(Spare, Trunk))$
$Goal(At(Spare, Axle))$
$Action(Remove(obj, loc),$
  PRECOND: $At(obj, loc)$
  EFFECT: $\neg At(obj, loc) \land At(obj, Ground))$
$Action(PutOn(t, Axle),$
  PRECOND: $Tire(t) \land At(t, Ground) \land \neg At(Flat, Axle)$
  EFFECT: $\neg At(t, Ground) \land At(t, Axle))$
$Action(LeaveOvernight,$
  PRECOND:
  EFFECT: $\neg At(Spare, Ground) \land \neg At(Spare, Axle) \land \neg At(Spare, Trunk)$
    $\land \neg At(Flat, Ground) \land \neg At(Flat, Axle) \land \neg At(Flat, Trunk))$

Example: *Changing the spare tire*

$Init(Tire(Flat) \wedge Tire(Spare) \wedge At(Flat, Axle) \wedge At(Spare, Trunk))$

$Goal(At(Spare, Axle))$

$Action(Remove(obj, loc),$

$\quad$ PRECOND: $At(obj, loc)$

$\quad$ EFFECT: $\neg At(obj, loc) \wedge At(obj, Ground))$

$Action(PutOn(t, Axle),$

$\quad$ PRECOND: $Tire(t) \wedge At(t, Ground) \wedge \neg At(Flat, Axle)$

$\quad$ EFFECT: $\neg At(t, Ground) \wedge At(t, Axle))$

$Action(LeaveOvernight,$

$\quad$ PRECOND:

$\quad$ EFFECT: $\neg At(Spare, Ground) \wedge \neg At(Spare, Axle) \wedge \neg At(Spare, Trunk)$
$\wedge \neg At(Flat, Ground) \wedge \neg At(Flat, Axle) \wedge \neg At(Flat, Trunk))$

Example: *Changing the spare tire*

$Init(Tire(Flat) \land Tire(Spare) \land At(Flat, Axle) \land At(Spare, Trunk))$
$Goal(At(Spare, Axle))$
$Action(Remove(obj, loc),$
  PRECOND: $At(obj, loc)$
  EFFECT: $\neg At(obj, loc) \land At(obj, Ground))$
$Action(PutOn(t, Axle),$
  PRECOND: $Tire(t) \land At(t, Ground) \land \neg At(Flat, Axle)$
  EFFECT: $\neg At(t, Ground) \land At(t, Axle))$
$Action(LeaveOvernight,$
  PRECOND:
  EFFECT: $\neg At(Spare, Ground) \land \neg At(Spare, Axle) \land \neg At(Spare, Trunk)$
  $\land \neg At(Flat, Ground) \land \neg At(Flat, Axle) \land \neg At(Flat, Trunk))$

Example: *Changing the spare tire*

$Init(Tire(Flat) \land Tire(Spare) \land At(Flat, Axle) \land At(Spare, Trunk))$
$Goal(At(Spare, Axle))$
$Action(Remove(obj, loc),$
  PRECOND: $At(obj, loc)$
  EFFECT: $\neg At(obj, loc) \land At(obj, Ground))$
$Action(PutOn(t, Axle),$
  PRECOND: $Tire(t) \land At(t, Ground) \land \neg At(Flat, Axle)$
  EFFECT: $\neg At(t, Ground) \land At(t, Axle))$
$Action(LeaveOvernight,$
  PRECOND:
  EFFECT: $\neg At(Spare, Ground) \land \neg At(Spare, Axle) \land \neg At(Spare, Trunk)$
  $\land \neg At(Flat, Ground) \land \neg At(Flat, Axle) \land \neg At(Flat, Trunk))$

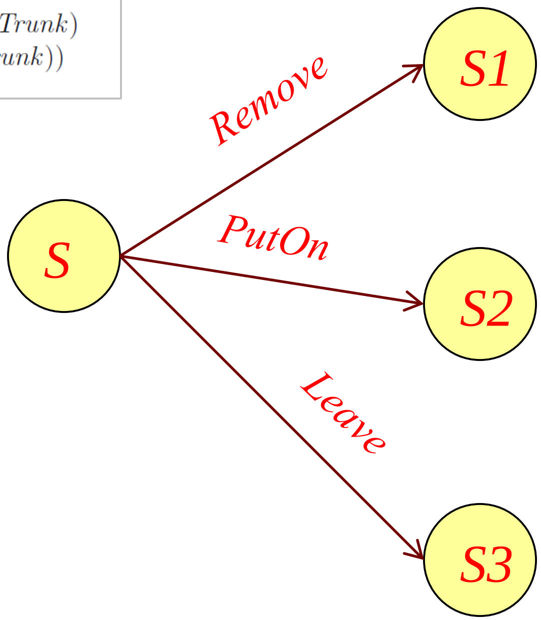
Example: *Changing the spare tire*

$Init(Tire(Flat) \land Tire(Spare) \land At(Flat, Axle) \land At(Spare, Trunk))$
$Goal(At(Spare, Axle))$
$Action(Remove(obj, loc),$
  PRECOND: $At(obj, loc)$
  EFFECT: $\neg At(obj, loc) \land At(obj, Ground))$
$Action(PutOn(t, Axle),$
  PRECOND: $Tire(t) \land At(t, Ground) \land \neg At(Flat, Axle)$
  EFFECT: $\neg At(t, Ground) \land At(t, Axle))$
$Action(LeaveOvernight,$
  PRECOND:
  EFFECT: $\neg At(Spare, Ground) \land \neg At(Spare, Axle) \land \neg At(Spare, Trunk)$
    $\land \neg At(Flat, Ground) \land \neg At(Flat, Axle) \land \neg At(Flat, Trunk))$

Bad neighborhood

Example: *Changing the spare tire*

$Init(Tire(Flat) \wedge Tire(Spare) \wedge At(Flat, Axle) \wedge At(Spare, Trunk))$
$Goal(At(Spare, Axle))$
$Action(Remove(obj, loc),$
  PRECOND: $At(obj, loc)$
  EFFECT: $\neg At(obj, loc) \wedge At(obj, Ground))$
$Action(PutOn(t, Axle),$
  PRECOND: $Tire(t) \wedge At(t, Ground) \wedge \neg At(Flat, Axle)$
  EFFECT: $\neg At(t, Ground) \wedge At(t, Axle))$
$Action(LeaveOvernight,$
  PRECOND:
  EFFECT: $\neg At(Spare, Ground) \wedge \neg At(Spare, Axle) \wedge \neg At(Spare, Trunk)$
  $\wedge \neg At(Flat, Ground) \wedge \neg At(Flat, Axle) \wedge \neg At(Flat, Trunk))$



A solution to the problem is $[Remove(Flat, Axle), Remove(Spare, Trunk), PutOn(Spare, Axle)]$.

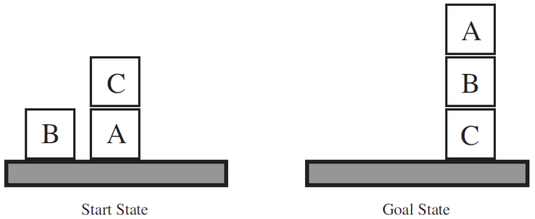
Example: *the blocks world*

Relations

- On(A, Table)
- On(B, Table)
- On(C, A)
- Clear (B)
- Clear (C)
- Block (A), Block (B), Block (C)

Start State   Goal State

Example: *the blocks world*



$$Init(On(A, Table) \wedge On(B, Table) \wedge On(C, A) \wedge Block(A) \wedge Block(B) \wedge Block(C) \wedge Clear(B) \wedge Clear(C))$$

$$\boxed{Goal(On(A, B) \wedge On(B, C))}$$

$Action(Move(b, x, y),$
  PRECOND: $On(b, x) \wedge Clear(b) \wedge Clear(y) \wedge Block(b) \wedge Block(y) \wedge (b \neq x) \wedge (b \neq y) \wedge (x \neq y),$
  EFFECT: $On(b, y) \wedge Clear(x) \wedge \neg On(b, x) \wedge \neg Clear(y))$

$Action(MoveToTable(b, x),$
  PRECOND: $On(b, x) \wedge Clear(b) \wedge Block(b) \wedge (b \neq x),$
  EFFECT: $On(b, Table) \wedge Clear(x) \wedge \neg On(b, x))$

Example: *the blocks world*

$$Init(On(A, Table) \land On(B, Table) \land On(C, A)$$
$$\land\ Block(A) \land Block(B) \land Block(C) \land Clear(B) \land Clear(C))$$

$$\boxed{Goal(On(A, B) \land On(B, C))}$$

$Action(Move(b, x, y),$

  PRECOND: $On(b, x) \land Clear(b) \land Clear(y) \land Block(b) \land Block(y) \land$
  $(b \neq x) \land (b \neq y) \land (x \neq y),$

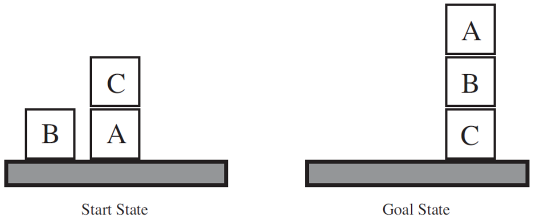
  EFFECT: $On(b, y) \land Clear(x) \land \neg On(b, x) \land \neg Clear(y))$

$Action(MoveToTable(b, x),$

  PRECOND: $On(b, x) \land Clear(b) \land Block(b) \land (b \neq x),$

  EFFECT: $On(b, Table) \land Clear(x) \land \neg On(b, x))$

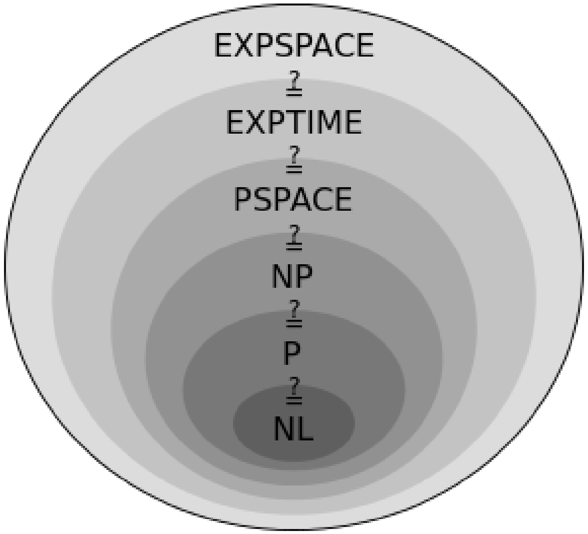
Example: *the blocks world*



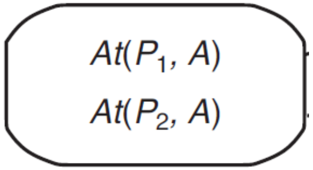
$$\begin{array}{l}
Init(On(A, Table) \wedge On(B, Table) \wedge On(C, A) \\
\quad \wedge\ Block(A) \wedge Block(B) \wedge Block(C) \wedge Clear(B) \wedge Clear(C)) \\
Goal(On(A, B) \wedge On(B, C)) \\
Action(Move(b, x, y), \\
\quad \text{PRECOND: } On(b, x) \wedge Clear(b) \wedge Clear(y) \wedge Block(b) \wedge Block(y) \wedge \\
\qquad\qquad (b \neq x) \wedge (b \neq y) \wedge (x \neq y), \\
\quad \text{EFFECT: } On(b, y) \wedge Clear(x) \wedge \neg On(b, x) \wedge \neg Clear(y)) \\
Action(MoveToTable(b, x), \\
\quad \text{PRECOND: } On(b, x) \wedge Clear(b) \wedge Block(b) \wedge (b \neq x), \\
\quad \text{EFFECT: } On(b, Table) \wedge Clear(x) \wedge \neg On(b, x))
\end{array}$$

Complexity of classic planning

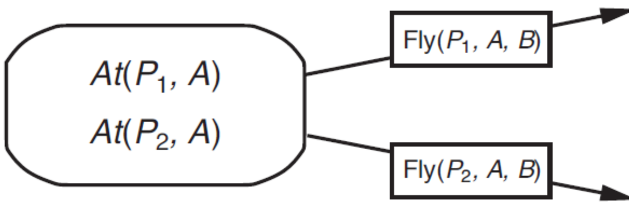
- PSPACE, a complexity class that is larger/harder than NP
  - **Planner**: ask for a sequence of actions that, if executed from a state, will make goal become true in a future state
    - PSPACE
  - **Theorem prover**: ask if a sentence is true given KB (does not have the notion of state transition)
    - NP



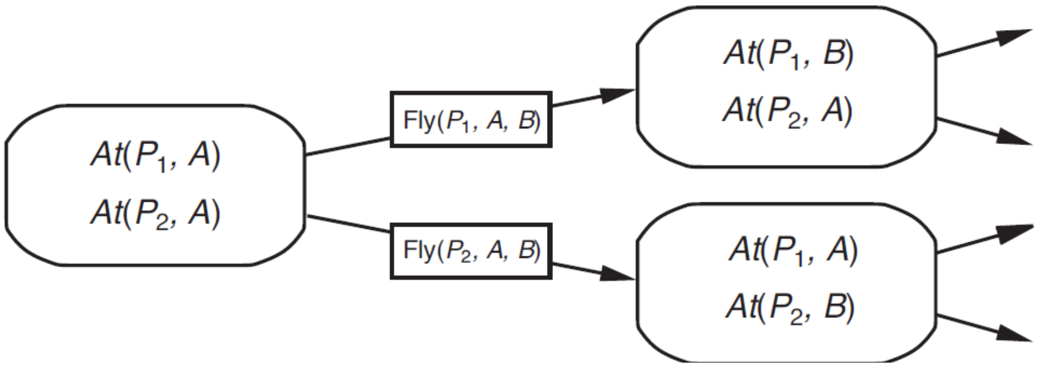
Planning as state-space search *(forward)*



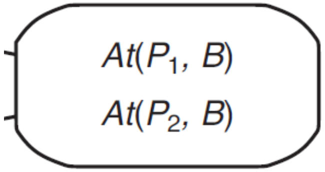
Planning as state-space search *(forward)*



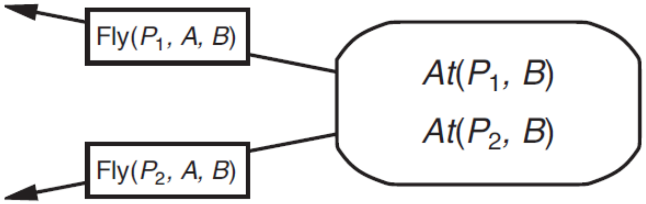
Planning as state-space search *(forward)*



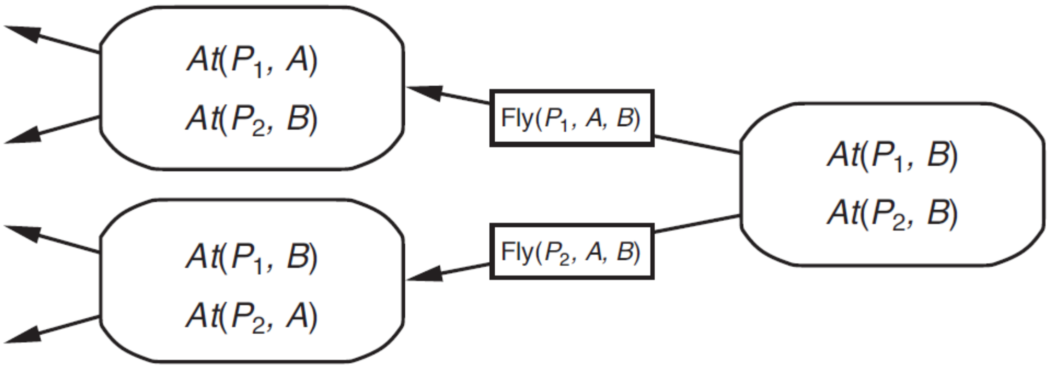
Planning as state-space search *(backward)*



Planning as state-space search *(backward)*



Planning as state-space search *(backward)*



Heuristics for planning

- Neither forward nor backward search is efficient without a good heuristic function
  - Need an admissible heuristic
  - i.e., never overestimate the distance from a state (s) to the goal

Planning graph

- It is a **data structure** used to give **heuristic estimates**
  - Can be applied to any of the search techniques
  - Will never overestimate; and often very accurate

$$
\begin{array}{l}
Init(Have(Cake)) \\
Goal(Have(Cake) \land Eaten(Cake)) \\
Action(Eat(Cake) \\
\quad \textsc{Precond}: Have(Cake) \\
\quad \textsc{Effect}: \neg Have(Cake) \land Eaten(Cake)) \\
Action(Bake(Cake) \\
\quad \textsc{Precond}: \neg Have(Cake) \\
\quad \textsc{Effect}: Have(Cake))
\end{array}
$$

Planning graph

- S0, S1, S2 – states
  - May be reachable at each level
  - Mutual exclusion (mutex) links
- A0, A1 – actions
  - Mutual exclusion (mutex) links

Init(*Have*(*Cake*))
Goal(*Have*(*Cake*) ∧ *Eaten*(*Cake*))
Action(*Eat*(*Cake*)
  PRECOND: *Have*(*Cake*)
  EFFECT: ¬ *Have*(*Cake*) ∧ *Eaten*(*Cake*))
Action(*Bake*(*Cake*)
  PRECOND: ¬ *Have*(*Cake*)
  EFFECT: *Have*(*Cake*))

S_0 A_0 S_1 A_1 S_2

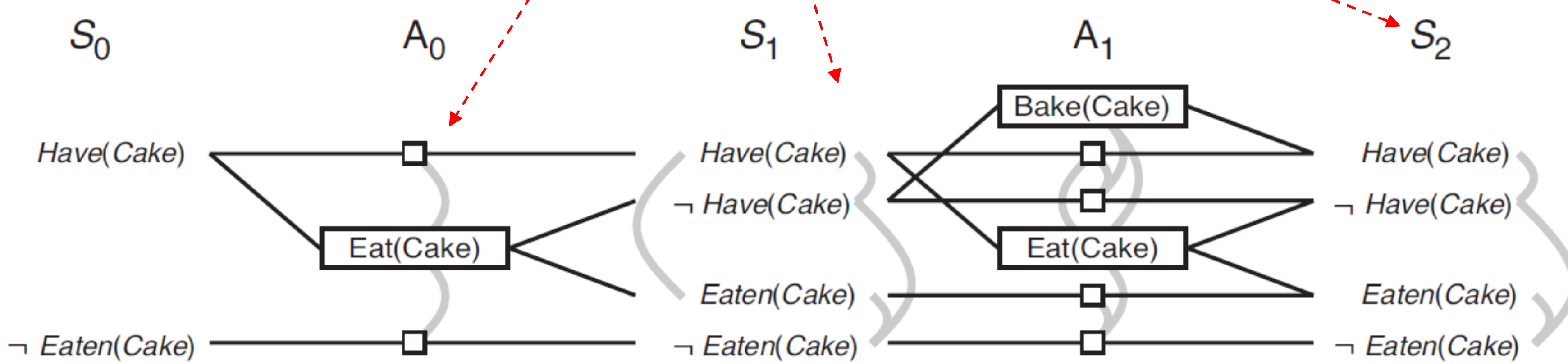
Have(*Cake*)

¬ *Eaten*(*Cake*)

Planning graph

- S0, S1, S2 – states
  - May be reachable at each level
  - Mutual exclusion (mutex) links
- A0, A1 – actions
  - Mutual exclusion (mutex) links

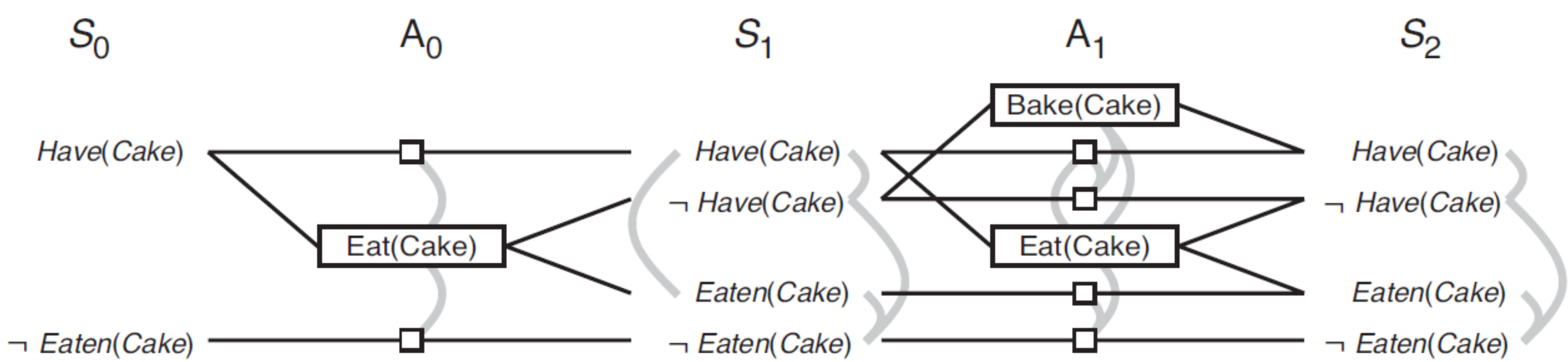
Init(*Have*(*Cake*))
Goal(*Have*(*Cake*) ∧ *Eaten*(*Cake*))
Action(*Eat*(*Cake*)
  PRECOND: *Have*(*Cake*)
  EFFECT: ¬ *Have*(*Cake*) ∧ *Eaten*(*Cake*))
Action(*Bake*(*Cake*)
  PRECOND: ¬ *Have*(*Cake*)
  EFFECT: *Have*(*Cake*))



Planning graph: *mutex actions*

- Effects contradict each other
  - Eat(Cake).effect vs. Have(Cake).effect
- Preconditions contradict each other
  - Bake(Cake).precond vs Eat(Cake).precond
- Interference (one action's effect contradicts the other action's precond)
  - Eat(Cake).effect vs Have(Cake).precond

Init(*Have*(*Cake*))
Goal(*Have*(*Cake*) ∧ *Eaten*(*Cake*))
Action(*Eat*(*Cake*)
  PRECOND: *Have*(*Cake*)
  EFFECT: ¬ *Have*(*Cake*) ∧ *Eaten*(*Cake*))
Action(*Bake*(*Cake*)
  PRECOND: ¬ *Have*(*Cake*)
  EFFECT: *Have*(*Cake*))



Properties of a planning graph

- **Polynomial** in the size of the planning problem
  – Instead of being "**exponential**" in size

- If any goal literal fails to appear in the final level of the graph, then the problem is **unsolvable**

- The **cost** of achieving any **goal (*g*)** can be **estimated** as the level at which (***g***) first appears in the planning graph constructed from (***s***) as the initial state

Graph planning algorithm

function GRAPHPLAN($problem$) **returns** solution or failure

  $graph \leftarrow$ INITIAL-PLANNING-GRAPH($problem$)
  $goals \leftarrow$ CONJUNCTS($problem$.GOAL)
  $nogoods \leftarrow$ an empty hash table
  for $tl = 0$ **to** ∞ **do**
    if $goals$ all non-mutex in S_t of $graph$ **then**
      $solution \leftarrow$ EXTRACT-SOLUTION($graph$, $goals$, NUMLEVELS($graph$), $nogoods$)
      if $solution \neq failure$ **then return** $solution$
    if $graph$ and $nogoods$ have both leveled off **then return** $failure$
    $graph \leftarrow$ EXPAND-GRAPH($graph$, $problem$)

Example planning graph (spare tire)

S_0 A_0 S_1 A_1 S_2

$At(Spare,Trunk)$

$At(Flat,Axle)$

$\neg At(Spare,Axle)$

$\neg At(Flat,Ground)$

$\neg At(Spare,Ground)$

Example planning graph (spare tire)