A Cut Into Classical Planning

Alain Mahieu, Dimitri Rusin

Knowledge-Based Systems Group
RWTH Aachen University
mail@domain.tld

ProSeminar KI

pics/logo$_{b}lau_{r}ec$

Outline

The Human wants the machine (agent) to imitate her.

Examples:

- Humans make coffee by following a step-by-step plan: Get a cup, press the big START button on the coffee machine, add some sugar, ...
- Humans travel on holiday by following a step-by-step plan: Buy a train ticket to Berlin *by July 17*, get on the train to Berlin *at 8 am*, check in at the hotel *by 12 am*, ...

The Human wants the machine (agent) to imitate her.

Examples:

- Humans make coffee by following a step-by-step plan: Get a cup, press the big START button on the coffee machine, add some sugar, ...
- Humans travel on holiday by following a step-by-step plan: Buy a train ticket to Berlin *by July 17*, get on the train to Berlin *at 8 am*, check in at the hotel *by 12 am*, ...

Outline

State-Transition System

State-Transition System

Σ=(S,A,E,γ)

- S is a set of states
- A is a set of actions
- E is a set of Events
- γ is a state-transition function

Restrictions

Classical planning deals with planning problems under certain restrictions:

Restrictions

- A1: S is a finite set of states
- A2: Σ is fully observable
- A3: Σ is deterministic
- A4: The planner handles only restricted goals

Restrictions

Classical planning deals with planning problems under certain restrictions:

Restrictions

- A1: S is a finite set of states
- A2: Σ is fully observable
- A3: Σ is deterministic
- A4: The planner handles only restricted goals

Restrictions

Classical planning deals with planning problems under certain restrictions:

Restrictions

- A1: S is a finite set of states
- A2: Σ is fully observable
- A3: Σ is deterministic
- A4: The planner handles only restricted goals

Restrictions

Classical planning deals with planning problems under certain restrictions:

Restrictions

- A1: S is a finite set of states
- A2: Σ is fully observable
- A3: Σ is deterministic
- A4: The planner handles only restricted goals

Restrictions

Restrictions

- A5: Σ is static
- A6: Sequential plans
- A7: Implicit time
- A8: Offline planning

Restrictions

Restrictions

- A5: Σ is static
- A6: Sequential plans
- A7: Implicit time
- A8: Offline planning

Restrictions

Restrictions

- A5: Σ is static
- A6: Sequential plans
- A7: Implicit time
- A8: Offline planning

Restrictions

Restrictions

- A5: Σ is static
- A6: Sequential plans
- A7: Implicit time
- A8: Offline planning

Planning Problem

Planning Problem

$P = (\Sigma, s_0, g)$

- Σ is the planning domain
- s_0 is the initial state
- g is the goal state or set of goal states

Problem: Is there a sequence of actions that will lead from the initial state to the goal state?

Outline

pics/logo$_b lau_r ec$

PDDL

PDDL

- Planning Domain Description Language
- Describes the parts of the state-transition system
- Provides a generalized description for planning Problems
- Main input language at International Planning Competitions

States

states

- Conjunction of fluents
- Fluents: ground, functionless atoms
- Unique names imply distinct objects

Example 1

$$at(r_1, l_1) \wedge loaded(r_1, c_1)$$

Robot r_1 is at location l_1 and loaded with cargo c_1

pics/logo$_{b}lau_{r}ec$

States

states

- Conjunction of fluents
- Fluents: ground, functionless atoms
- Unique names imply distinct objects

Example 1

$$at(r_1, l_1) \wedge loaded(r_1, c_1)$$

Robot r_1 is at location l_1 and loaded with cargo c_1

pics/logo$_{b}lau_{r}ec$

States

states

- Conjunction of fluents
- Fluents: ground, functionless atoms
- Unique names imply distinct objects

Example 1

$$at(r_1, l_1) \wedge loaded(r_1, c_1)$$

Robot r_1 is at location l_1 and loaded with cargo c_1

States

states

- Conjunction of fluents
- Fluents: ground, functionless atoms
- Unique names imply distinct objects

Example 1

$$at(r_1, l_1) \wedge loaded(r_1, c_1)$$

Robot r_1 is at location l_1 and loaded with cargo c_1

Action schemas

Action Schemas

- List of object names
- Preconditions
- Effects

Action move

$Action(move(r_1, l_1, l_2)$

$Precondition : at(r_1, l_1) \wedge robot(r_1) \wedge location(l_1) \wedge location(l_2)$

$Effect : \neg at(r_1, l_1) \wedge at(r_1, l_2)$

Action schemas

Action Schemas

- List of object names
- Preconditions
- Effects

Action move

$Action(move(r_1, l_1, l_2)$
$Precondition : at(r_1, l_1) \wedge robot(r_1) \wedge location(l_1) \wedge location(l_2)$
$Effect : \neg at(r_1, l_1) \wedge at(r_1, l_2)$

Example

Example: Cargo-transport

$Init(at(r_1, l_1) \wedge at(c_1, l_1) \wedge robot(r_1) \wedge cargo(c_1) \wedge location(l_1) \wedge location(l_2))$

$Goal(at(c_1, l_2))$

Example

$Action(load(r, c, l)$

$Precon : at(r, l) \wedge at(c, l) \wedge robot(r) \wedge cargo(c) \wedge loaction(l)$
$Effect : \neg at(c, l) \wedge loaded(r, c))$

$Action(unload(r, c, l)$

$Precon : at(r, l) \wedge loaded(r, c) \wedge robot(r) \wedge cargo(c) \wedge location(l)$
$Effect : \neg loaded(r, c) \wedge at(c, l))$

$Action(move(r, from, to)$

$Precon : at(r, from) \wedge robot(r) \wedge location(from) \wedge location(to)$
$Effect : \neg at(r, from) \wedge at(r, to)$

Example

- Actions affect predicates

Solution-Plan

$load(r_1, c_1, l_1)$
$move(r_1, l_1, l_2)$
$unload(r_1, c_1, l_2)$

Outline

SAT-Problem

SAT-Problem

Given a formula φ.
Does there exist a model that satisfies φ?

steps to follow

- Translate the classical planning problem into a satisfiability problem
- Determine if there exists a plan by solving the satisfiability problem with a satisfiability decision procedure.
- Extract the plan from the assignments determined by the satisfiability decision procedure

states

- still ground,functionless atoms
- additionally negations are allowed

example

$at(r_1, l_1) \wedge loaded(r_1, c_1)$

solutions with second location

$$\mu_1 = (at(r_1, l_1) \leftarrow true, loaded(r_1, c_1) \leftarrow true, at(r_1, l_2) \leftarrow false)$$
$$\mu_2 = (at(r_1, l_1) \leftarrow true, loaded(r_1, c_1) \leftarrow true, at(r_1, l_2) \leftarrow true)$$

pics/logo$_b lau_r ec$

state-transitions

- Add a timestep i to every predicate
- An action itself is a predicate now
- The action implies preconditions and effects

Example

$$move(r_1, l_1, l_2, s_1) \Rightarrow$$
$$(at(r_1, l_1, s_1) \wedge \neg at(r_1, l_2, s_1) \wedge \neg at(r_1, l_1, s_2) \wedge at(r_1, l_2, s_2))$$

Formulas

Formulas

- Initial state: $\bigwedge_{f \in s0} f_0 \wedge \bigwedge_{f \notin s_0} \neg f_0$
- Goal state: $\bigwedge_{f \in g^+} f_n \wedge \bigwedge_{f \in g^-} f_n$
- Actions: $a_i \Rightarrow \left(\bigwedge_{p \in precond(a)} p_i \wedge \bigwedge_{e \in effects(a)} e_{i+1} \right)$

pics/logo$_b lau_r ec$

Formulas

Formulas

- Initial state: $\bigwedge_{f \in s0} f_0 \wedge \bigwedge_{f \notin s_0} \neg f_0$
- Goal state: $\bigwedge_{f \in g^+} f_n \wedge \bigwedge_{f \in g^-} f_n$
- Actions: $a_i \Rightarrow \left(\bigwedge_{p \in precond(a)} p_i \wedge \bigwedge_{e \in effects(a)} e_{i+1} \right)$

Formulas

Formulas

- Initial state: $\bigwedge_{f \in s0} f_0 \wedge \bigwedge_{f \notin s_0} \neg f_0$
- Goal state: $\bigwedge_{f \in g^+} f_n \wedge \bigwedge_{f \in g^-} f_n$
- Actions: $a_i \Rightarrow \left(\bigwedge_{p \in precond(a)} p_i \wedge \bigwedge_{e \in effects(a)} e_{i+1} \right)$

pics/logo$_{b}lau_{r}ec$

Formulas

Formulas

- Explanatory frame axioms:
  $$\left(\neg f_i \wedge f_{i+1} \Rightarrow \left(\bigvee_{a \in A | f_i \in \mathit{effects}^+(a)} a_i\right)\right) \bigwedge$$
  $$\left(f_i \wedge \neg f_{i+1} \Rightarrow \left(\bigvee_{a \in A | f_i \in \mathit{effects}^-(a)} a_i\right)\right)$$
- Complete exclusion axioms: $\neg a_i \vee \neg b_i$

pics/logo$_b lau_r ec$

Formulas

Formulas

- Explanatory frame axioms:
$$\left(\neg f_i \wedge f_{i+1} \Rightarrow \left(\bigvee_{a \in A | f_i \in effects^+(a)} a_i\right)\right) \bigwedge$$
$$\left(f_i \wedge \neg f_{i+1} \Rightarrow \left(\bigvee_{a \in A | f_i \in effects^-(a)} a_i\right)\right)$$
- Complete exclusion axioms: $\neg a_i \vee \neg b_i$

Example

Example: Cargo-transport

Init-Goal

- $(init) \quad at(r_1, l_1, 0) \land at(c_1, l_1, 0) \land \neg at(r_1, l_2, 0) \land \ldots$
- $(goal) \quad at(c_1, l_2, 3) \land \neg at(r_1, l_1, 2) \land \ldots$

Actions

- $move1 \quad move(r_1, l_1, l_2, 0) \Rightarrow$
  $(at(r_1, l_1, 0) \land at(r_1, l_2, 1) \land \neg at(r_1, l_1, 1))$
- $move2 \quad move(r_1, l_2, l_1, 0) \Rightarrow$
  $(at(r_1, l_2, 0) \land at(r_1, l_1, 1) \land \neg at(r_1, l_2, 1))$
- $\ldots$

Example

Example: Cargo-transport

Explanatory frame axioms

- $\neg at(r_1, l_1, 0) \wedge at(r_1, l_1, 1) \Rightarrow move(r_1, l_2, l_1, 0)$
- $\neg at(r_1, l_2, 0) \wedge at(r_1, l_2, 1) \Rightarrow move(r_1, l_1, l_2, 0)$
- $at(r_1, l_1, 0) \wedge \neg at(r_1, l_1, 1) \Rightarrow move(r_1, l_1, l_2, 0)$
- $at(r_1, l_2, 0) \wedge \neg at(r_1, l_2, 1) \Rightarrow move(r_1, l_2, l_1, 0)$
- $\ldots$

Example

Example: Cargo-transport

Complete exclusion axioms

- $\neg move(r_1, l_1, l_2, 0) \vee \neg move(r_1, l_2, l_1, 0)$
- ...

Solution

- $load(r_1, c_1, l_1, 0), move(r_1, l_1, l_2, 1), unload(r_1, c_1, l_2, 2)$

Outline

pics/logo$_b lau_r ec$

Outline

The Planning Graph is Graphplan's main data structure.

- Layered graph in which one layer corresponds to one time step in the plan
- Layer i consists of one set of actions that are applicable at layer i and one set of literals that *could* be true at layer i.
- For every positive and negative literal p, we add the persistence action α_p with precondition p and effect p.

Forward Graph Expansion vs. Backward Search

Mutex Links

Literals

Actions

Goals

Backtracking

Forward Graph Expansion vs. Backward Search

Forward Graph Expansion

Mutex Links

Literals

Actions

Goals

Backtracking

Forward Graph Expansion vs. Backward Search

Forward Graph Expansion	Backward Search
Mutex Links	
Literals	
	Goals
Actions	Backtracking

Outline

pics/logo$_{b}lau_{r}ec$

The planning graph is expanded up to the point that it might be possible to achieve the goal.

Notation:

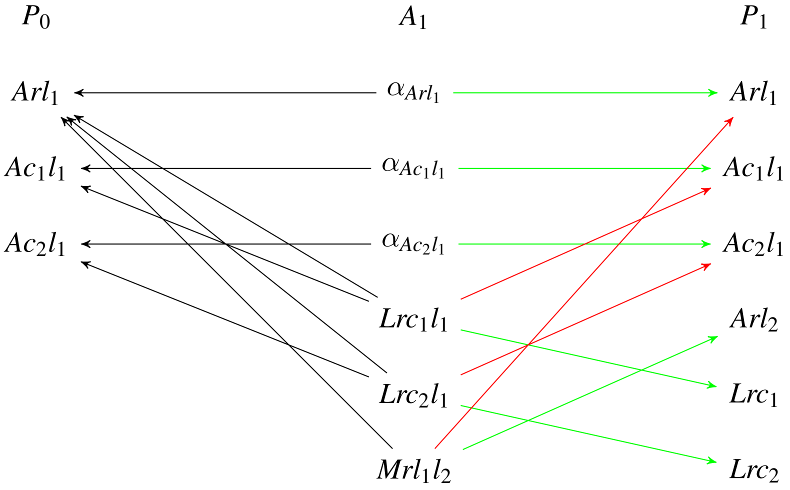
Literals: $at(r, l_1) \longrightarrow Arl_1$,
$at(c_1, l_2) \longrightarrow Ac_1l_2$,
$loaded(r, c_1) \longrightarrow Lrc_1$,
etc...

Actions: $move(r, l_1, l_2) \longrightarrow Mrl_1l_2$,
$unload(r, c_1, l_1) \longrightarrow Urc_1l_1$,
etc...

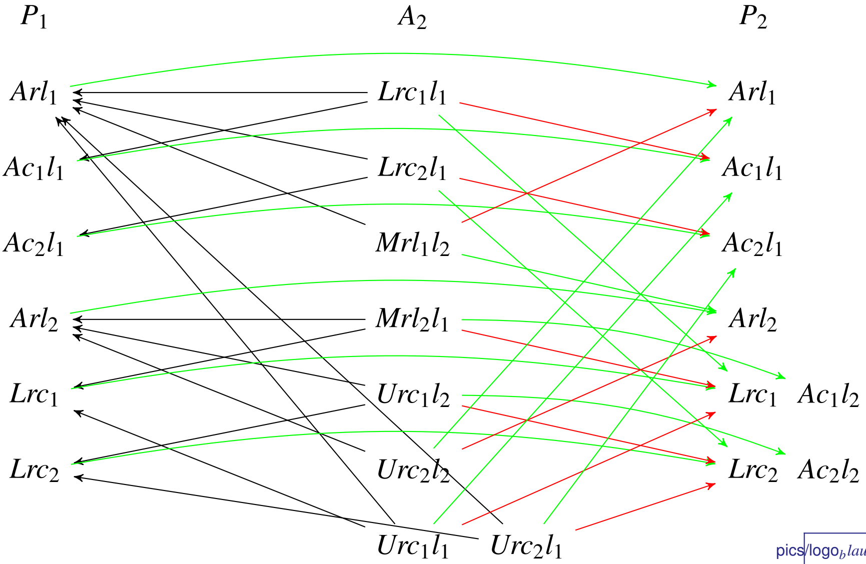
Cargo Transport (CT) problem:

$Init(Arl_1, Ac_1l_1, Ac_2l_1)$
$Goal(Ac_1l_2, Ac_2l_2)$

The planning graph is expanded up to the point that it might be possible to achieve the goal.

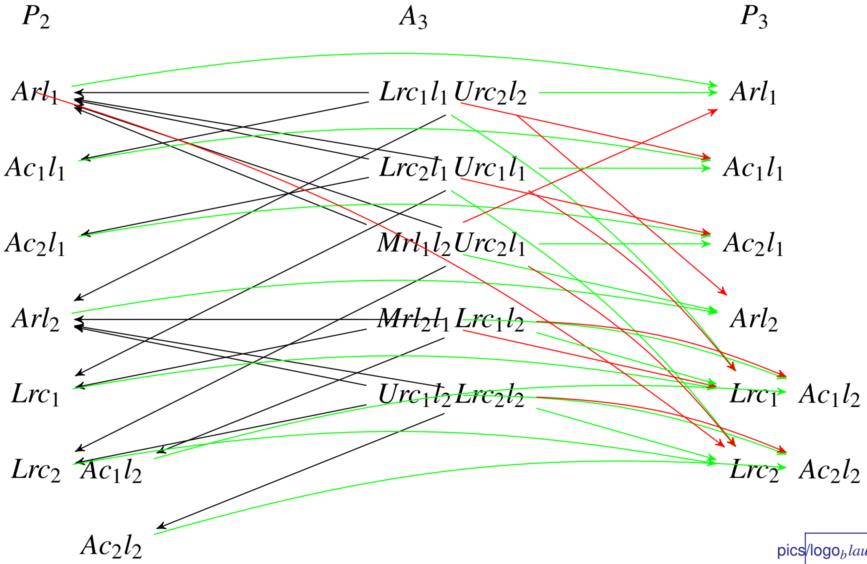


The planning graph is expanded up to the point that it might be possible to achieve the goal.



pics/logo$_b lau_r ec$

The planning graph is expanded up to the point that it might be possible to achieve the goal.



Mutex Links indicate whether two literals or actions are incompatible with each other.

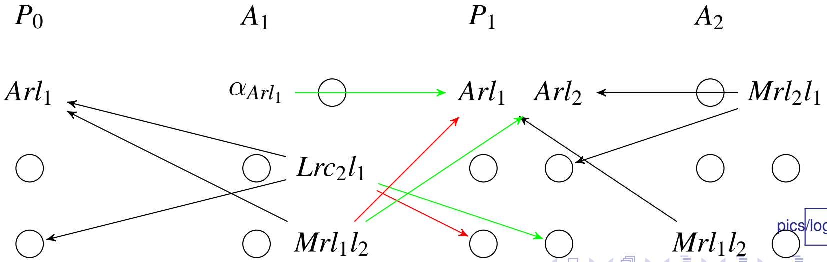
Mutex Links for Actions:

Two actions a_1 and a_2 are mutex in P_i if and only if (iff) one of the following conditions holds:

- *Interference:* action a_1 deletes a positive effect or a precondition of action a_2 or vice versa *(Dependence)*
- *Competing Needs:* precondition of a_1 is mutex with a precondition with a_2 or vice versa

Notation: $(a_1, a_2) \in \mu A_i$

Example (not all mutex links shown):



Mutex Links indicate whether two literals or actions are incompatible with each other.

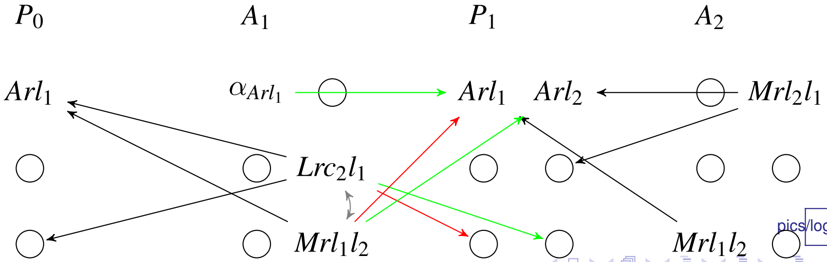
Mutex Links for Actions:

Two actions a_1 and a_2 are mutex in P_i if and only if (iff) one of the following conditions holds:

- *Interference:* action a_1 deletes a positive effect or a precondition of action a_2 or vice versa *(Dependence)*
- *Competing Needs:* precondition of a_1 is mutex with a precondition with a_2 or vice versa

Notation: $(a_1, a_2) \in \mu A_i$

Example (not all mutex links shown):



Mutex Links indicate whether two literals or actions are incompatible with each other.

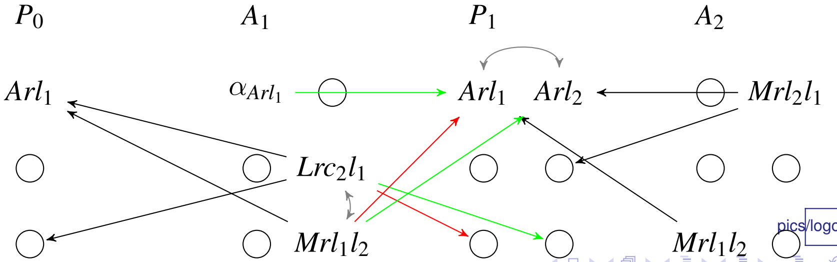
Mutex Links for Actions:

Two actions a_1 and a_2 are mutex in P_i if and only if (iff) one of the following conditions holds:

- *Interference:* action a_1 deletes a positive effect or a precondition of action a_2 or vice versa *(Dependence)*
- *Competing Needs:* precondition of a_1 is mutex with a precondition with a_2 or vice versa

Notation: $(a_1, a_2) \in \mu A_i$

Example (not all mutex links shown):



Mutex Links indicate whether two literals or actions are incompatible with each other.

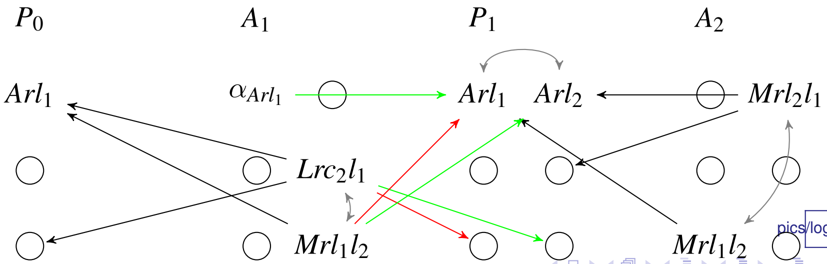
Mutex Links for Actions:

Two actions a_1 and a_2 are mutex in P_i if and only if (iff) one of the following conditions holds:

- *Interference:* action a_1 deletes a positive effect or a precondition of action a_2 or vice versa *(Dependence)*
- *Competing Needs:* precondition of a_1 is mutex with a precondition with a_2 or vice versa

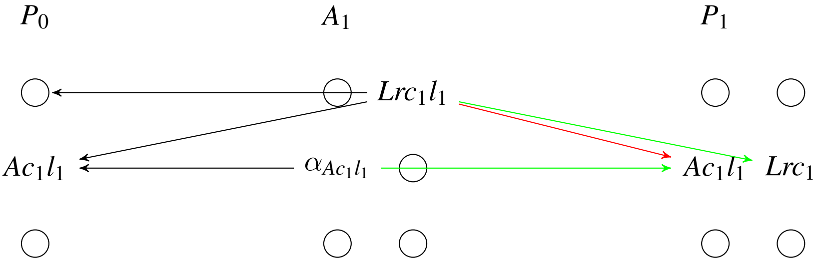
Notation: $(a_1, a_2) \in \mu A_i$

Example (not all mutex links shown):



Mutex Links indicate whether two literals or actions are incompatible with each other.

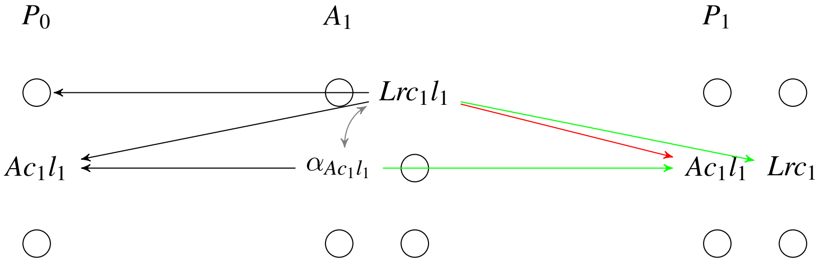
Mutex Links for Literals:
Two literals p_1 and p_2 are mutex in P_i iff no nonmutex pair of actions (a_1, a_2) exists such that a_1 produces p_1 and a_2 produces p_2.
Notation: $(p_1, p_2) \in \mu P_i$. We set $\mu P_0 = \emptyset$.
Example:



pics/logo$_b lau_r ec$

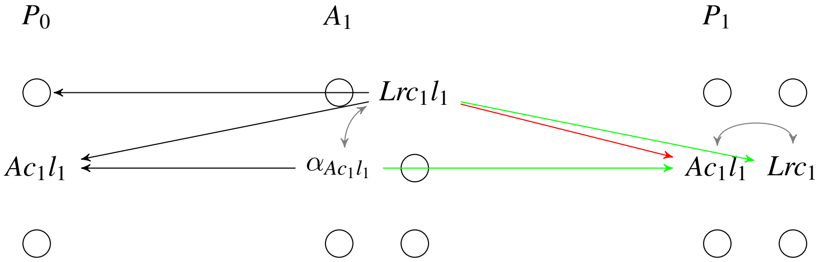
Mutex Links indicate whether two literals or actions are incompatible with each other.

Mutex Links for Literals:
Two literals p_1 and p_2 are mutex in P_i iff no nonmutex pair of actions (a_1, a_2) exists such that a_1 produces p_1 and a_2 produces p_2.
Notation: $(p_1, p_2) \in \mu P_i$. We set $\mu P_0 = \emptyset$.
Example:



Mutex Links indicate whether two literals or actions are incompatible with each other.

Mutex Links for Literals:
Two literals p_1 and p_2 are mutex in P_i iff no nonmutex pair of actions (a_1, a_2) exists such that a_1 produces p_1 and a_2 produces p_2.
Notation: $(p_1, p_2) \in \mu P_i$. We set $\mu P_0 = \emptyset$.
Example:



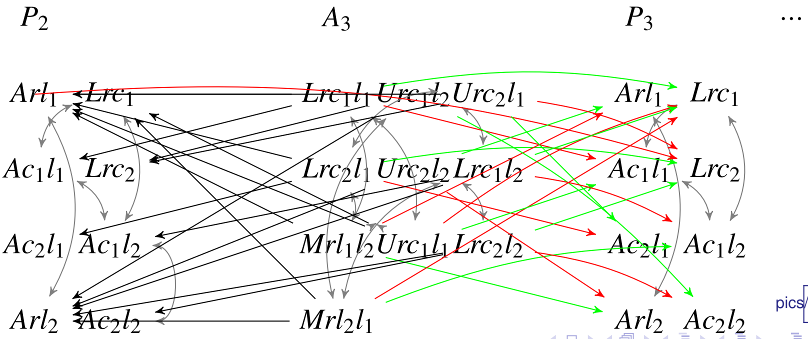
pics/logo$_b lau_r ec$

When a planning graph does not grow anymore, it has a reached its fixed point.

Given any planning graph G. Then there is a smallest k such that all of the following conditions hold:

- *Fixed Proposition Layer:* number of propositions at layer k is equal to number of propositions at any layer $i > k$
- *Fixed Prop. Mutex Links:* number of mutual links at layer k is equal to number of mutual links at any layer $i > k$

Example:

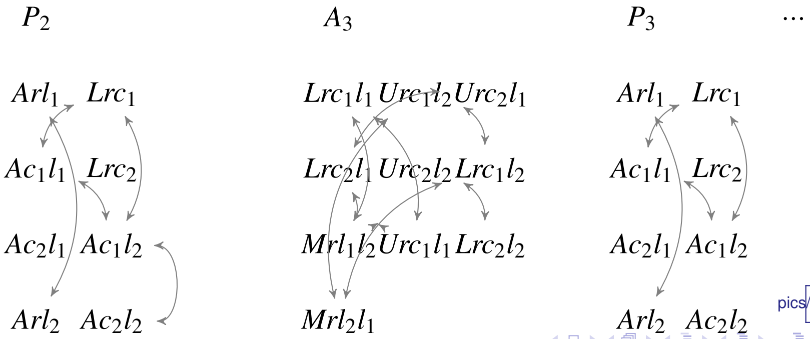


When a planning graph does not grow anymore, it has a reached its fixed point.

Given any planning graph G. Then there is a smallest k such that all of the following conditions hold:

- *Fixed Proposition Layer:* number of propositions at layer k is equal to number of propositions at any layer $i > k$
- *Fixed Prop. Mutex Links:* number of mutual links at layer k is equal to number of mutual links at any layer $i > k$

Example:

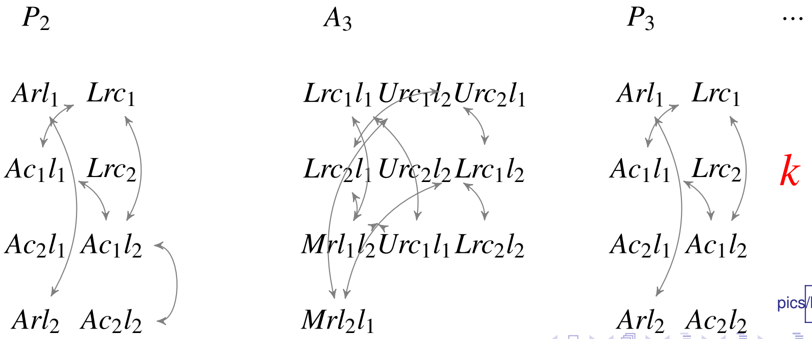


When a planning graph does not grow anymore, it has a reached its fixed point.

Given any planning graph G. Then there is a smallest k such that all of the following conditions hold:

- *Fixed Proposition Layer:* number of propositions at layer k is equal to number of propositions at any layer $i > k$
- *Fixed Prop. Mutex Links:* number of mutual links at layer k is equal to number of mutual links at any layer $i > k$

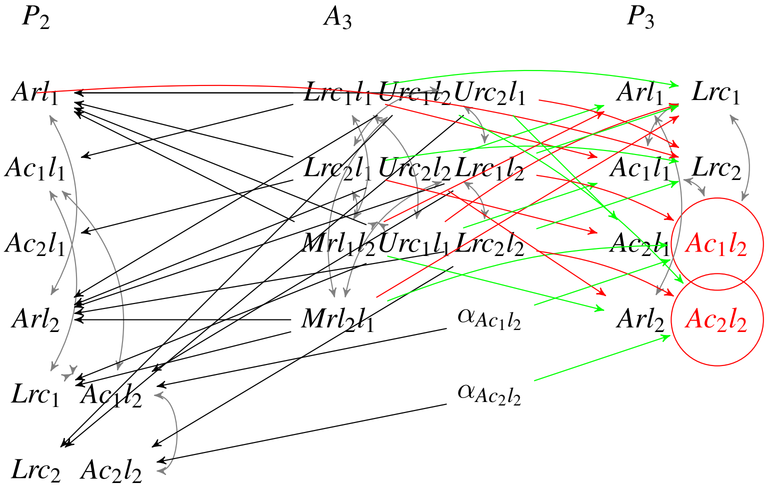
Example:



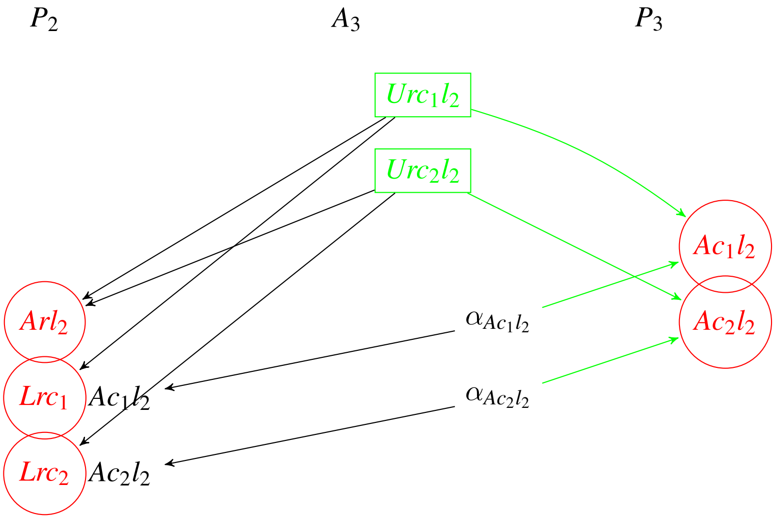
$k = 3$

Outline

Graphplan recursively tracks the goal set down to the initial state.

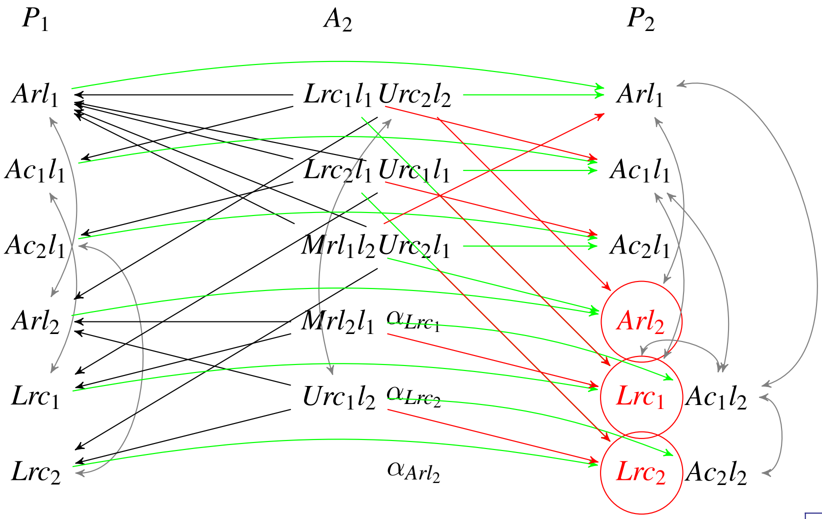


Graphplan recursively tracks the goal set down to the initial state.



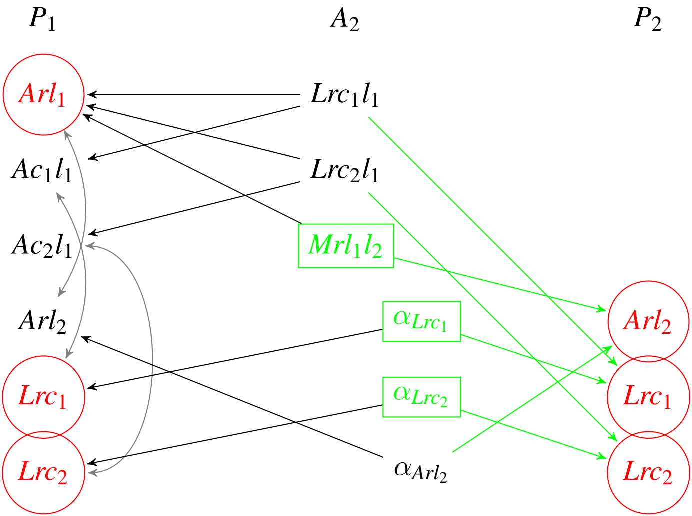
pics/logo$_{b}lau_{r}ec$

Graphplan recursively tracks the goal set down to the initial state.

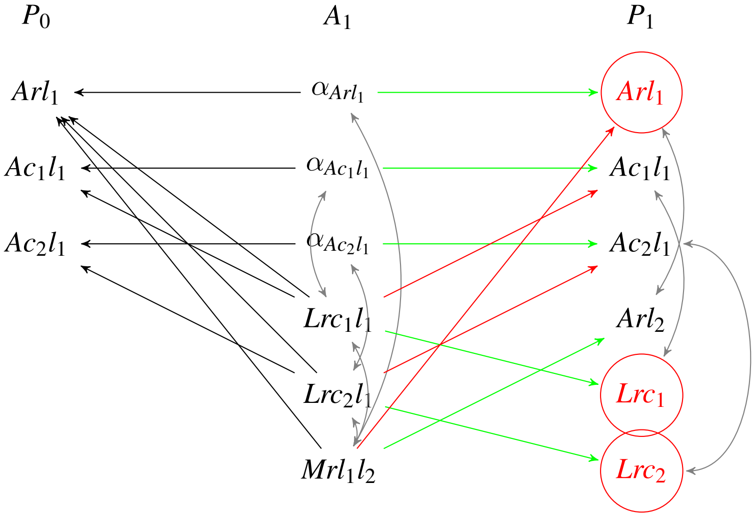


pics/logo$_b lau_r ec$

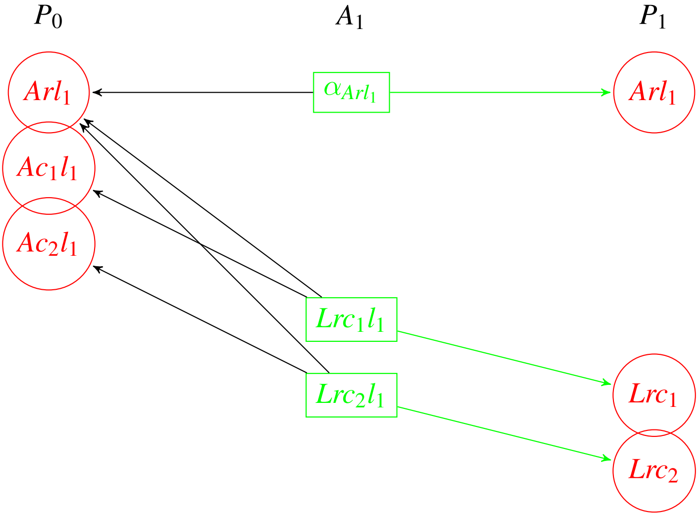
Graphplan recursively tracks the goal set down to the initial state.



Graphplan recursively tracks the goal set down to the initial state.



Graphplan recursively tracks the goal set down to the initial state.



pics/logo$_b lau_r ec$

Graphplan recursively tracks the goal set down to the initial state.

Layered Solution Plan:

- contains sets of nonmutex actions that correspond to a specific layer in the graph
- has a strict order of elements, i.e. each set of nonmutex actions has to performed sequentially (but the order within one set is arbitrary)
- the layered solution plan to the example problem instance: $\Pi = (\pi_1, \pi_2, \pi_3) = (\{Lrc_1l_1, Lrc_2l_1\}, \{Mrl_1l_2\}, \{Urc_1l_2, Urc_2l_2\})$.

Graphplan and SATPlan have assets and drawbacks.

	Asset	Drawback
Graphplan	▸ data structure	▸ no good heuristics (e.g. for choosing producers in BS)
SATPlan	▸ permanently new SAT solvers coming up, hot research	▸ number of clauses might be unfeasible to ground

CP systems are deployed in critical situations.

- Satellite launch
- Hubble Space Telescope
- etc...

pics/logo$_b lau_r ec$