Planificación Automática

Escuela de Verano de Inteligencia Artificial

Eva Onaindía

3 Septiembre 2014

Main reference:

Malik Ghallab, Dana S. Nau, Paolo Traverso
Automated planning - theory and practice.
Elsevier 2004, ISBN 978-1-55860-856-6, pp. I-XXVIII, 1-635

Acknowledgements

Some of the slides used in this course are taken from Dana Nau's lecture slides for the textbook *Automated Planning,* licensed under the Creative Commons Attribution-NonCommercial-ShareAlike License:

http://creativecommons.org/licenses/by-nc-sa/2.0/
http://creativecommons.org/licenses/by/3.0/es/

I would like to gratefully acknowledge Dana Nau's contributions and thank him for generously permitting me to use aspects of his presentation material.

Outline

1. The problem of planning in Artificial Intelligence (AI)
2. State of the art in automated planning
3. Beyond classical planning

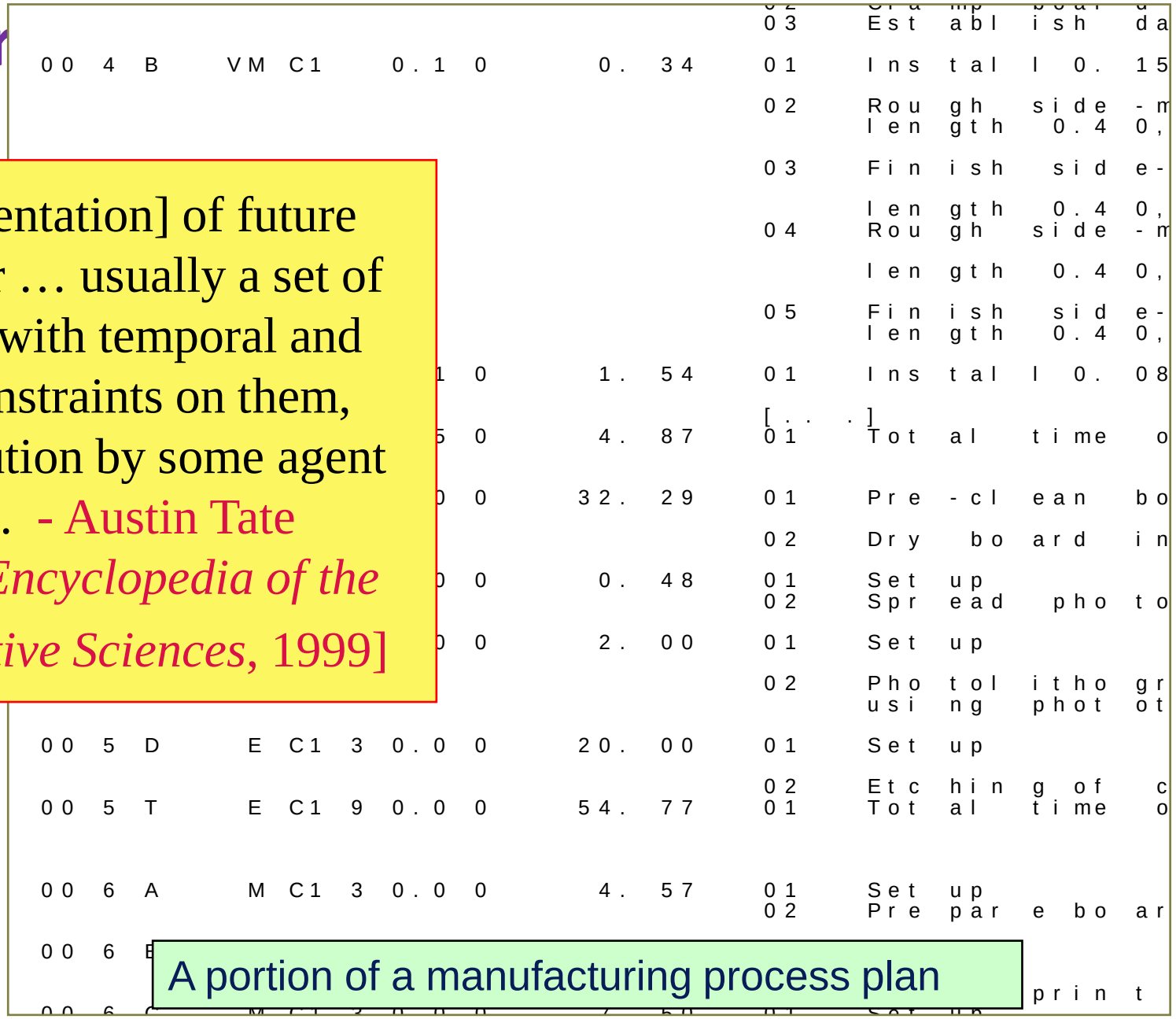
1. The pr

pl

1.

2.

[a representation] of future behavior … usually a set of actions, with temporal and other constraints on them, for execution by some agent or agents. - Austin Tate [*MIT Encyclopedia of the Cognitive Sciences*, 1999]



								03	Est	abl ish da
004	B	VM	C1	0.10		0.34	01	Ins	tal l 0. 15	
							02	Rou	gh side -m	
								len	gth 0.4 0,	
							03	Fin	ish side -	
								len	gth 0.4 0,	
							04	Rou	gh side -m	
								len	gth 0.4 0,	
							05	Fin	ish side -	
								len	gth 0.4 0,	
				1 0		1.54	01	Ins	tal l 0. 08	
							[. . .]			
				5 0		4.87	01	Tot	al time o	
				0 0		32.29	01	Pre	-cl ean bo	
							02	Dry	boa rd in	
				0 0		0.48	01	Set	up	
							02	Spr	ead pho to	
				0 0		2.00	01	Set	up	
							02	Pho	tol itho gr	
								usi	ng phot ot	
005	D	E	C1	3 0.0 0		20.00	01	Set	up	
							02	Etc	hin g of c	
005	T	E	C1	9 0.0 0		54.77	01	Tot	al time o	
006	A	M	C1	3 0.0 0		4.57	01	Set	up	
							02	Pre	par e bo ar	
006	B									
										print
006	C	M	C1	3 0.0 0		4.50	01	Set	up	

A portion of a manufacturing process plan

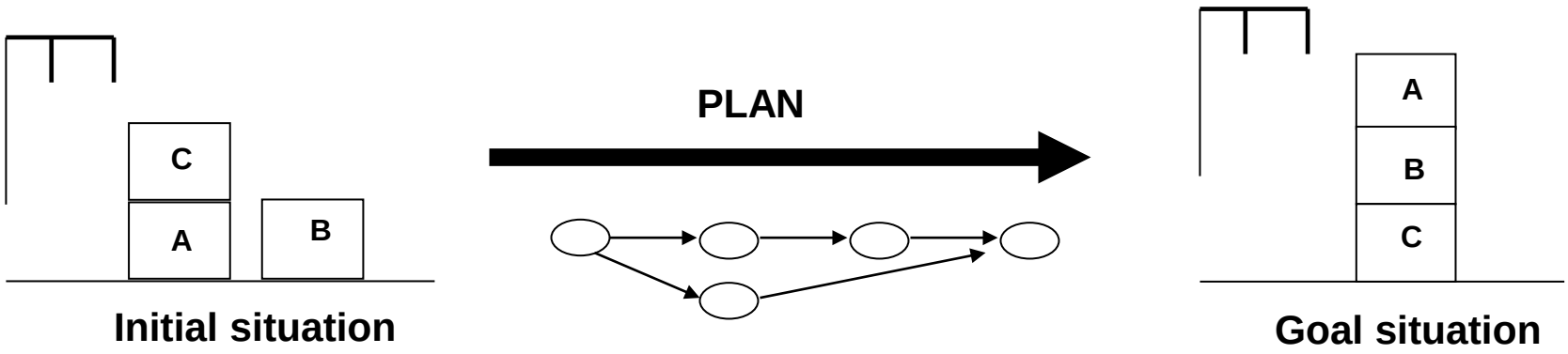
1. The problem of planning in AI

Planning problem

Input: initial state, goal state and applicable actions (operators) in the domain

Output: a plan: a set of partially ordered actions

Example: blocks-world domain



- Applicable operators:
  - **pick-up** a block from the table
  - **unstack** a block from another block
  - **stack** a block on top of another block
  - **drop** a holding block on the table

PLAN:

Time step 1: unstack (C,A)

Time step 2: drop (C)

Time step 3: pick-up (B)

Time step 4: stack (B,C)

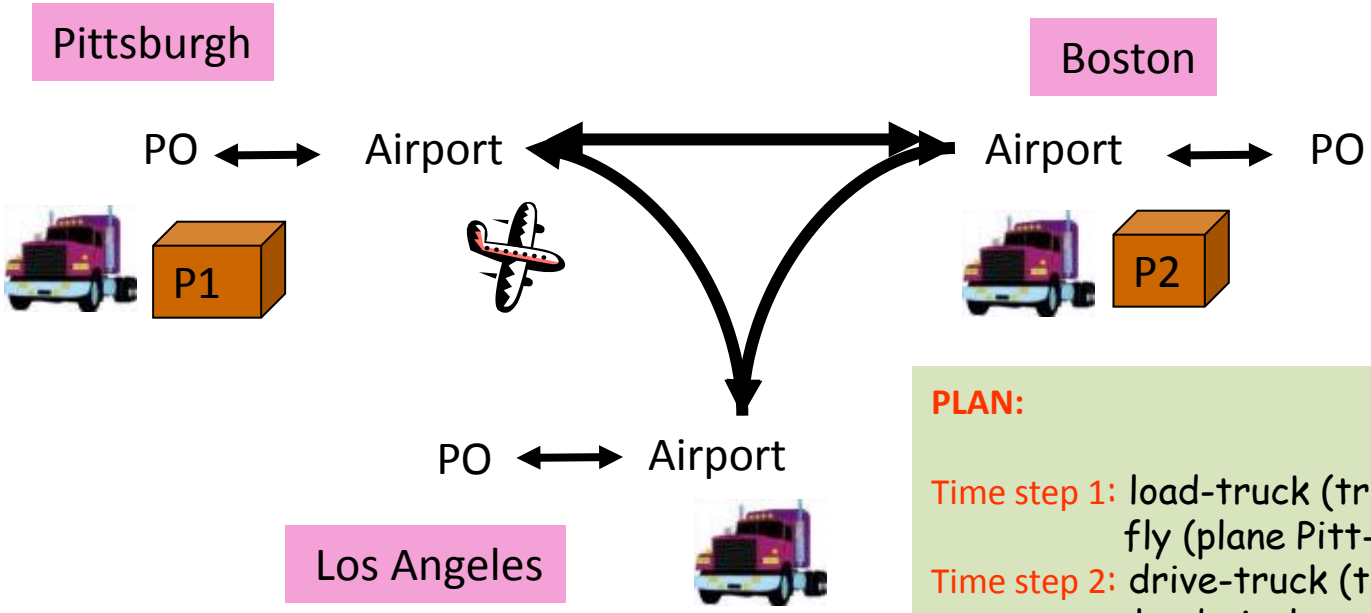
Time step 5: pick-up (A)

Time step 6: stack (A,B)

1. The problem of planning in AI

Example: Logistics domain

operators: load-truck, unload-truck, drive, load-airplane, unloadairplane, fly



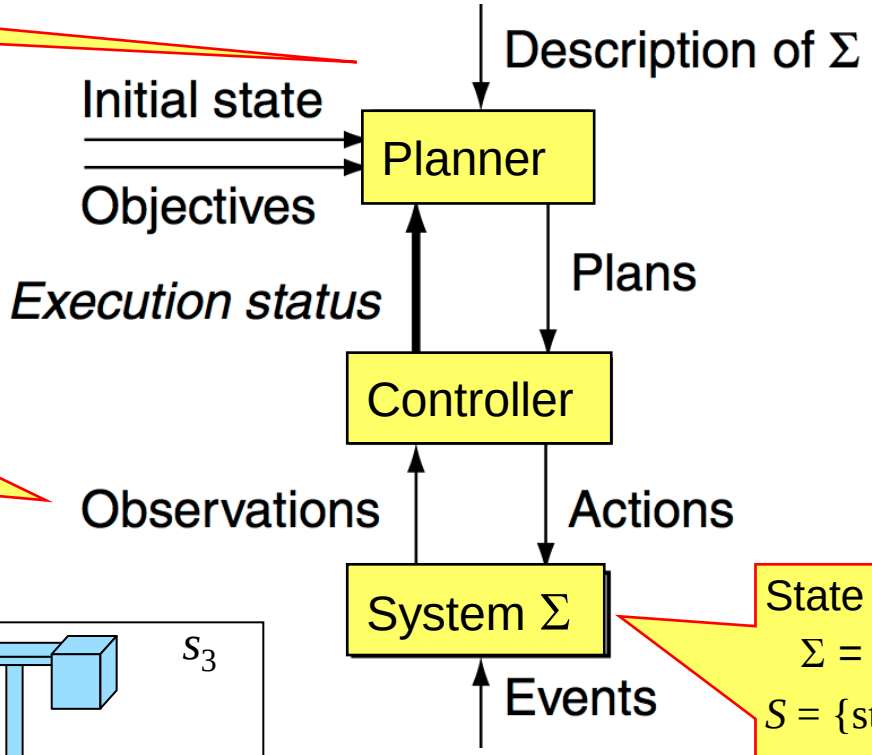
PLAN:

Time step 1: load-truck (tr-Pitt P1 Pitt-PO)
fly (plane Pitt-Air Bost-Air)
Time step 2: drive-truck (tr-Pitt Pitt-PO Pitt-Air)
load-airplane (plane P2 Bost-Air)
Time step 3: unload-truck (tr-Pitt P1 Pitt-Air)
fly (plane Bost-Air LA-Air)
Time step 4: unload-airplane (plane P2 LA-Air)
Time step 5: fly (plane LA-Air Pitt-Air)
Time step 6: load-airplane (plane P1 Pitt-Air)
Time step 7: fly (plane Pitt-Air LA-Air)
Time step 8: unload-airplane (plane P1 LA-Air)

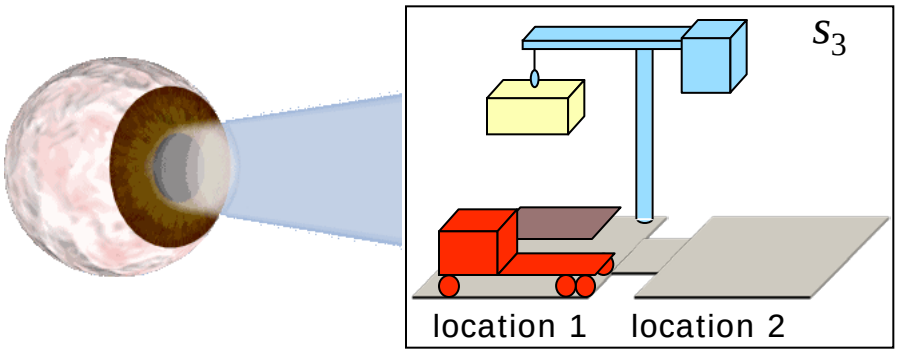
1. The problem of planning in AI

Conceptual model. Environment. Controller. Planner.

Planning problem



Observation function
$h: S \rightarrow O$



State transition system
$\Sigma = (S,A,E,\gamma)$
S = {states}
A = {actions}
E = {exogenous events}
γ = state-transition function

1. The problem of planning in AI

Conceptual model. Environment. Controller. Planner.

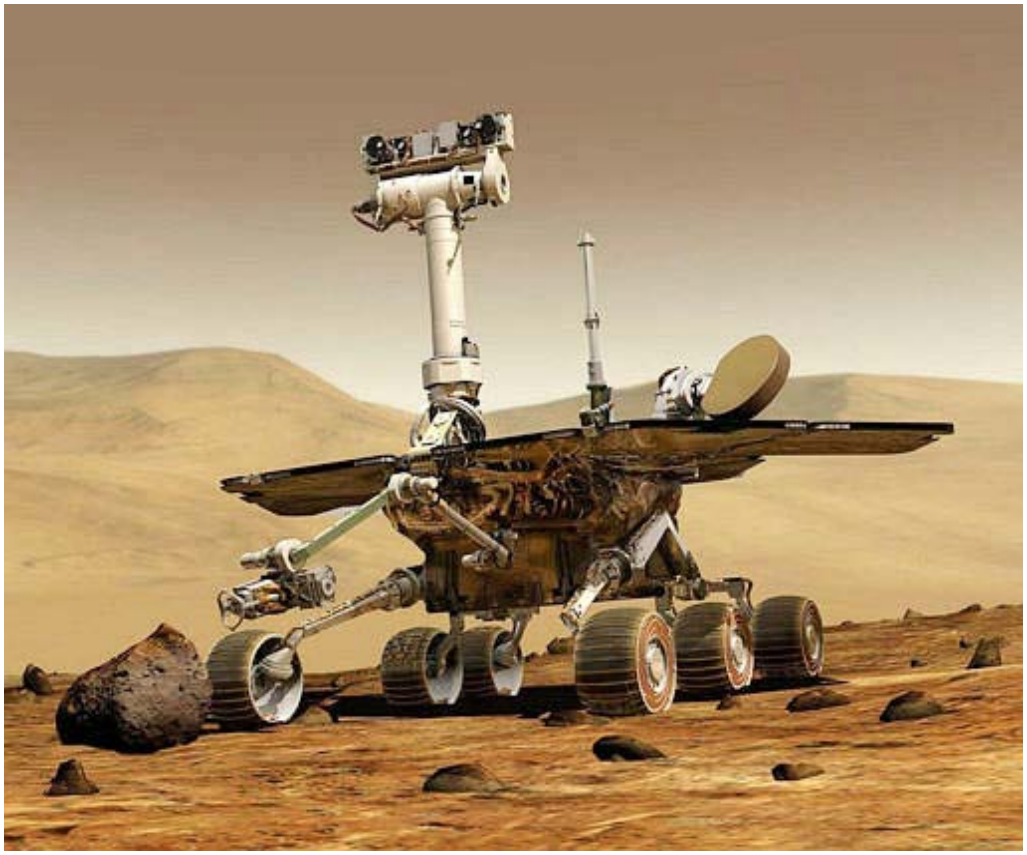
VIDEO

Blocks world with low-level simulation.
Sussman anomaly with a plan failure when picking up block B

1. The problem of planning in AI

Planning in real-world problems

- Autonomous planning, scheduling, control
  - NASA: JPL and Ames
- Remote Agent Experiment (RAX)
  - Deep Space 1
- Mars Exploration Rover (MER)



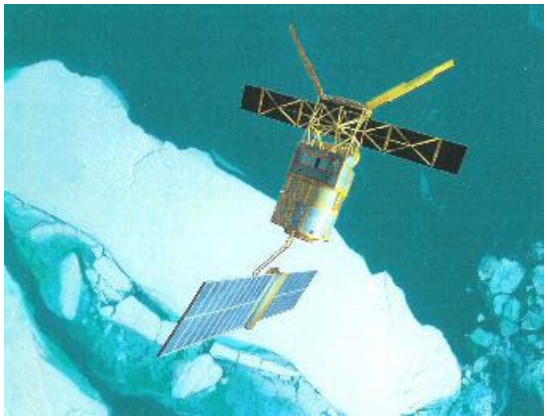
1. The problem of planning in AI

Planning in real-world problems

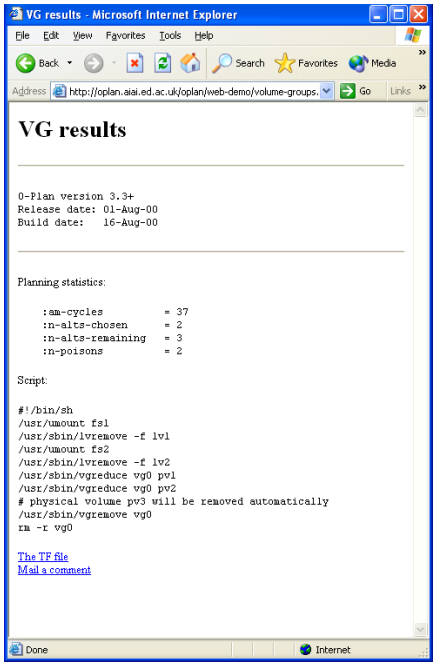
O-Plan
Artificial Intelligence Applications Institute, University of Edinburgh
http://www.aiai.ed.ac.uk/project/plan/

Evacuation operation
Rescue missions
Emergency response

Spacecraft Mission Planning (Optimum-AIV)
Unmanned Autonomous Vehicles



Systems Management Aids
Business Management Tasks
Help Desks and Assistants



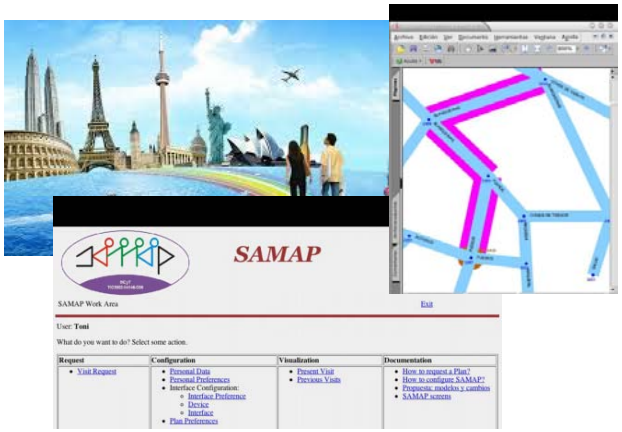
1. The problem of planning in AI

Planning in real-world problems

Forest fire fighting (SIADEX , UGR)
http://siadex.ugr.es/

E-learning:
http://adenu.ia.uned.es/adaptaplan/

Planning tourist visits:
SAMAP
http://iactive.es (nativoo)



Robotics

2. State of the art in planning

Classical planning requires all eight of the restrictive assumptions:

A0: Finite
A1: Fully observable
A2: Deterministic
A3: Static
A4: Attainment goals
A5: Sequential plans
A6: Implicit time
A7: Offline planning

2. State of the art in planning

Representation in planning

Planning Domain Description Language (PDDL):

- logic-based language
- popularized by the International Planning Competitions (IPC)

PDDL functionalities:

- object-type hierarchy
- constant objects
- predicates: (on ?x – block ?y – block) ➔ (on A B)
  (in ?x – package ?y – truck) ➔ (in P1 t1)
- numeric fluents (fuel ?x – plane) (energy ?r- rover)
  (time-to-walk ?l1 ?l2 – location)
- derived predicates
- timed initial literals
- object fluents (on-block ?x - block) – block ➔ (= (on-block B) A)
  (position ?x – package) – (either location truck) ➔ (= (position P1) t1)
- state-trajectory constraints (hard constraints in form of the modal-logic expressions)
- preferences (soft constraints)
- …

2. State of the art in planning

Representation in planning

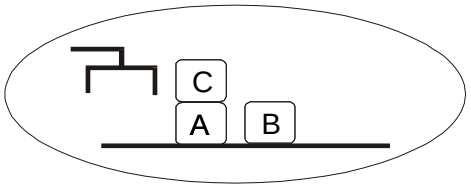
```
(:action stack
  :parameters (?ob – block ?underob - block)
  :precondition
     (and (clear ?underob)(holding ?ob))
  :effect
     (and (clear ?ob) (arm-empty) (on ?ob ?underob)
          (not (holding ?ob))(not (clear ?underob))))
```

```
(:action DRIVE-TRUCK
  :parameters (?truck - truck ?loc-from - place ?loc-to - place ?city - city)
  :precondition
   (and (at ?truck ?loc-from) (in-city ?loc-from ?city) (in-city ?loc-to ?city))
  :effect
   (and (at ?truck ?loc-to)) (not (at ?truck ?loc-from)))
```

2. State of the art in planning

Representation in planning

state: set of ground atoms (facts, literals, propositions, fluents)



(on C A) (on A table) (on B table)
(clear C)(clear B)(arm-empty)

action: An action a is *applicable* to a state s if s satisfies $\text{precond}(a)$,
i.e., if $\text{precond}^+(a) \subseteq s$ and $\text{precond}^-(a) \cap s = \varnothing$

```
(:action pickup
  :parameters (?ob - block)
  :precondition
     (and (clear ?ob)(on-table ?ob)(arm-empty))
  :effect
     (and (holding ?ob) (not (clear ?ob))
          (not (on-table ?ob))(not (arm-empty))))
```

The result of applying an action a in state s: (1) Remove a's negative effects, and (2) add a's positive effects:

$$\gamma(s,a) = (s - \text{effects}^-(a)) \cup \text{effects}^+(a)$$

2. State of the art in planning

Representation in planning

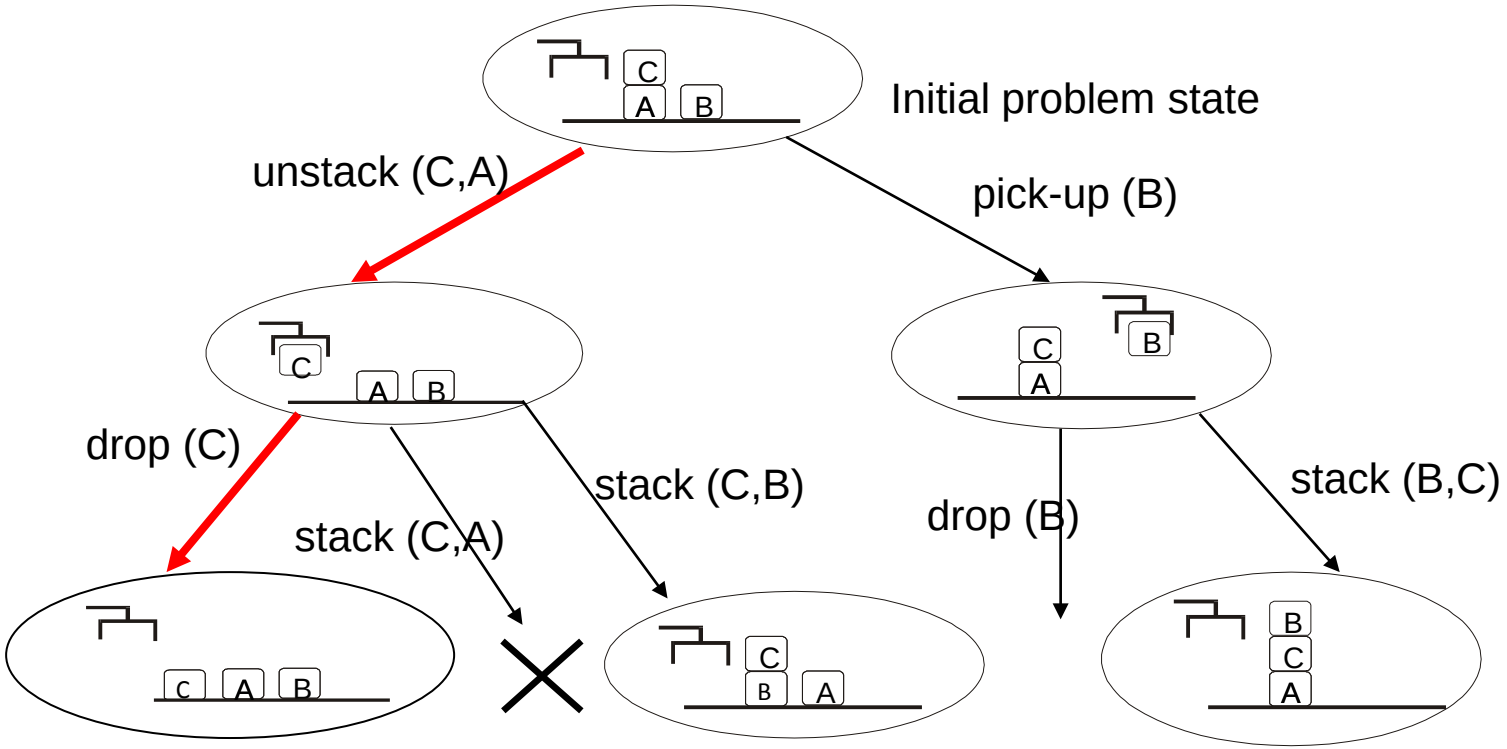
Formally, a planning task is a tuple $P=\langle O, s_0, G\rangle$ where:

- O is the collection of operators
- s_0 is a state (the initial state)
- G is a set of literals (the goal formula)

2. State of the art in planning

Planning techniques: total-order planning

- precursor: situation calculus
- state-based representation
- total-order planning => plans are sequences of totally ordered action
- planning order = execution order
- forward-chaining/backward-chaining reasoning



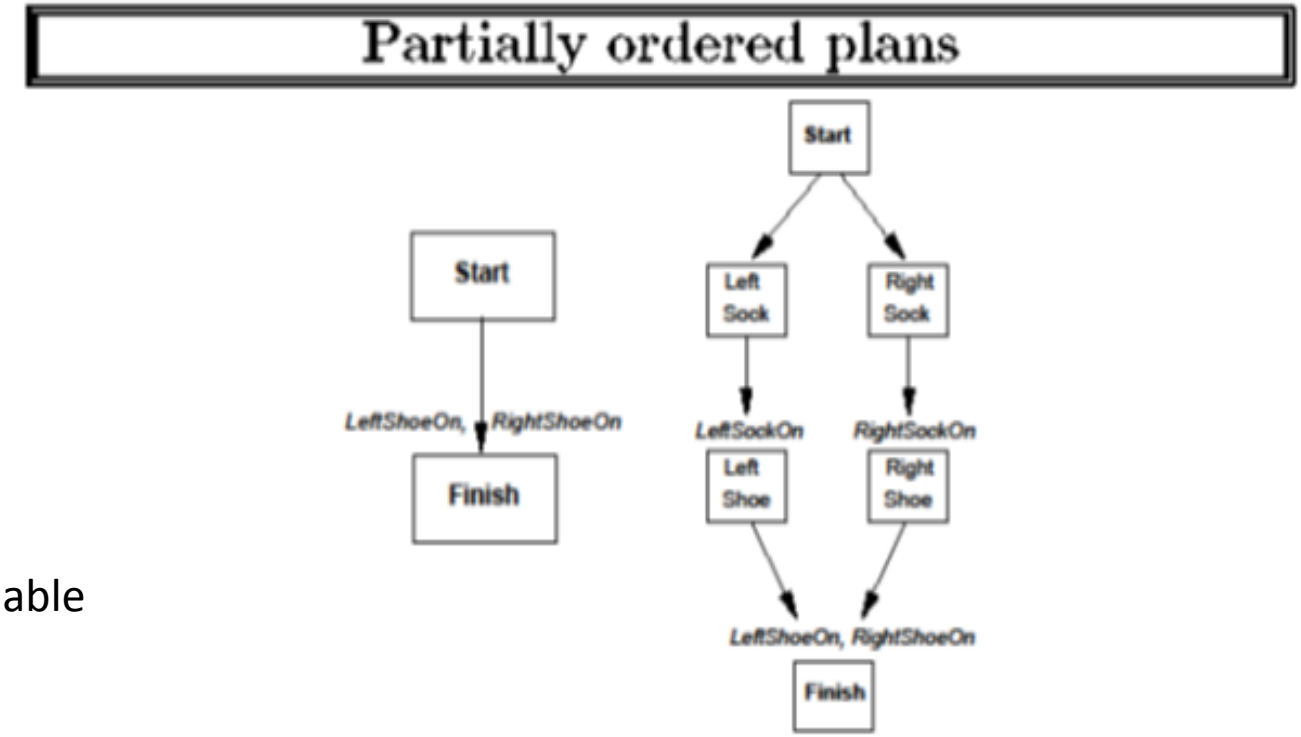
2. State of the art in planning

Planning techniques: partial-order planning (POP)

- plan-based representation
- plans are sets of partially ordered actions
- planning order ≠ execution order
- backward-chaining reasoning
- least-commitmment strategy

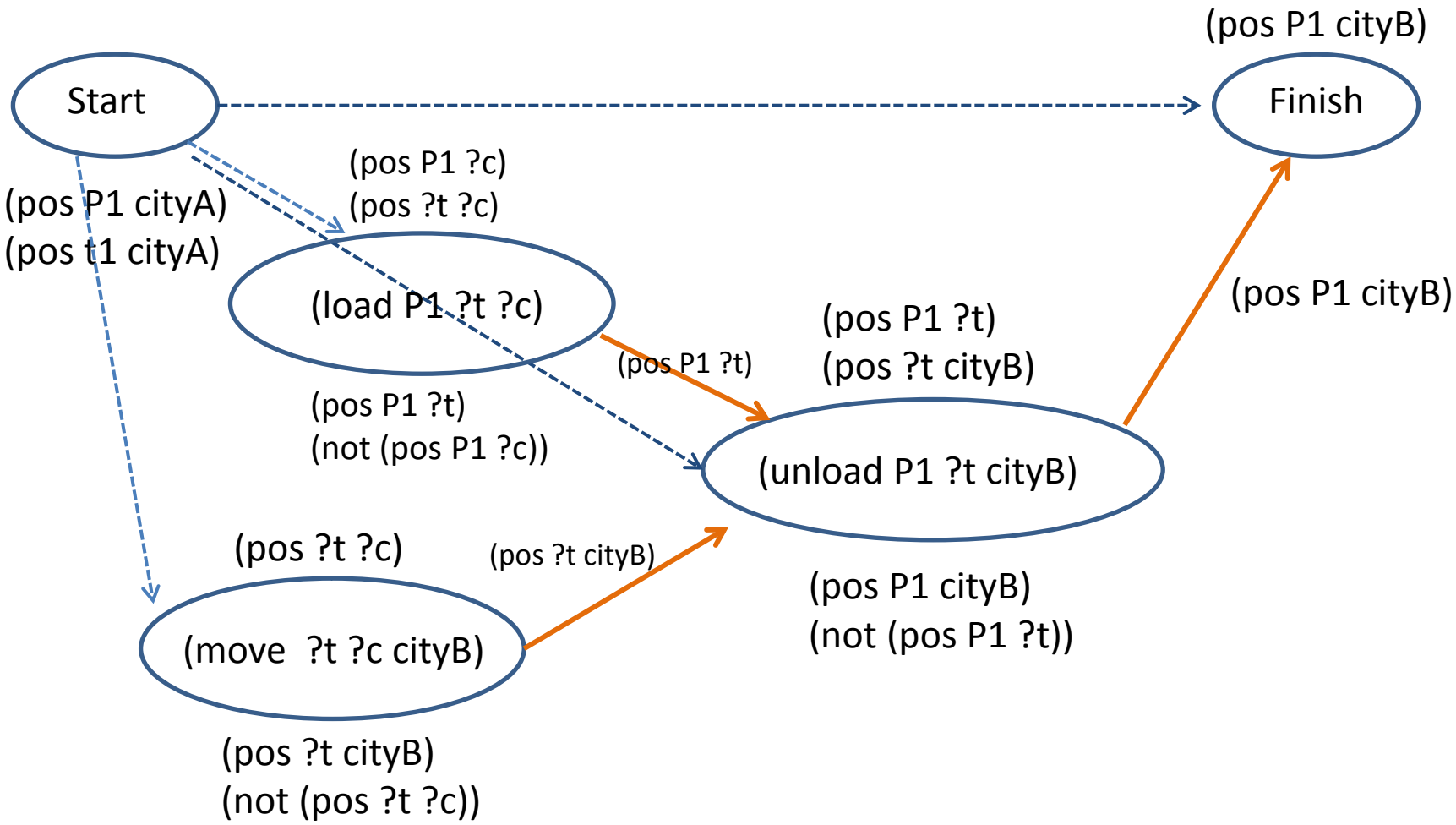
Constraints:
causal-link
ordering constraint
binding constraint

Flaws:
open goal
non-instantiated variable
threat



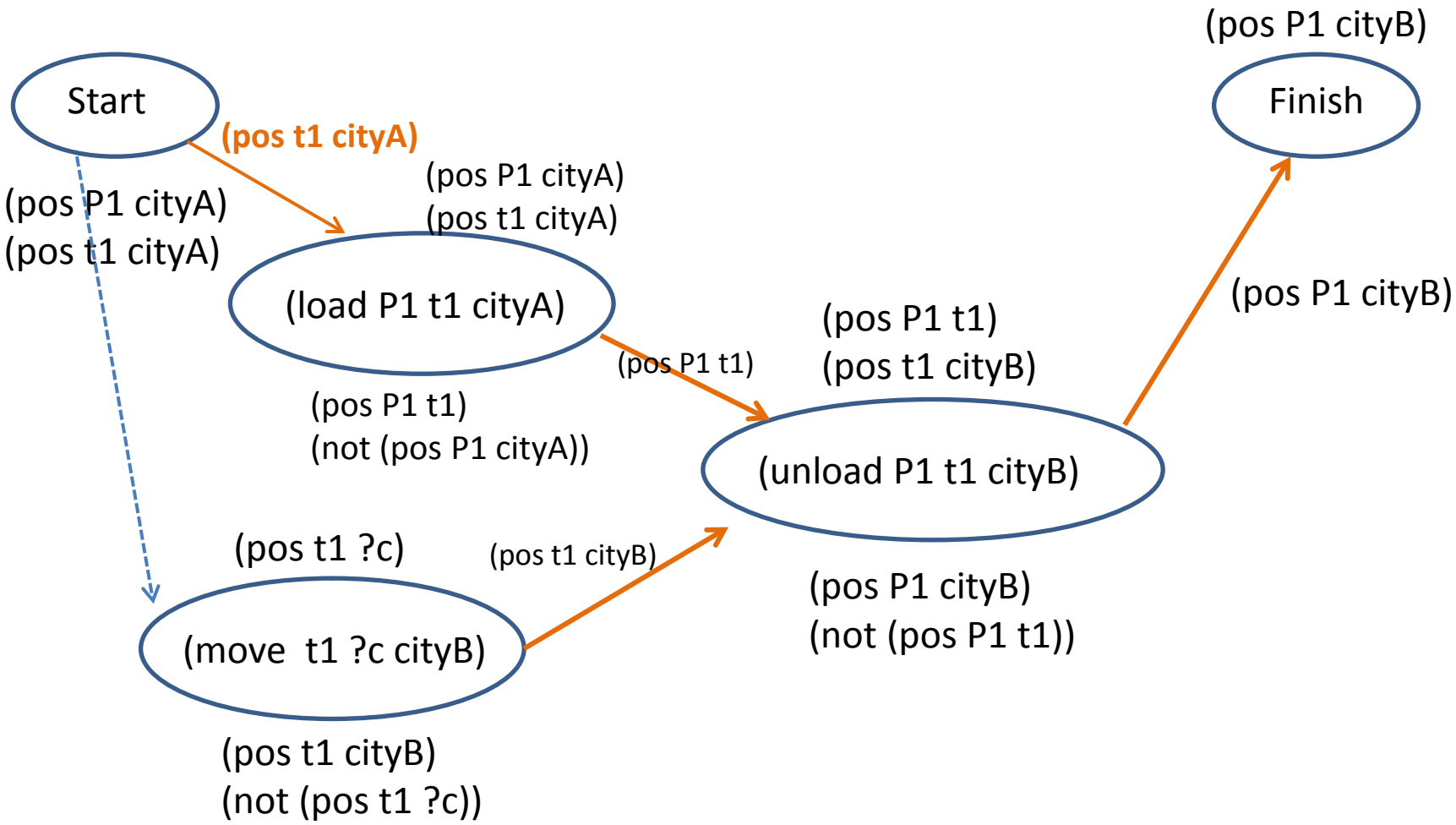
2. State of the art in planning

Planning techniques: partial-order planning (POP)



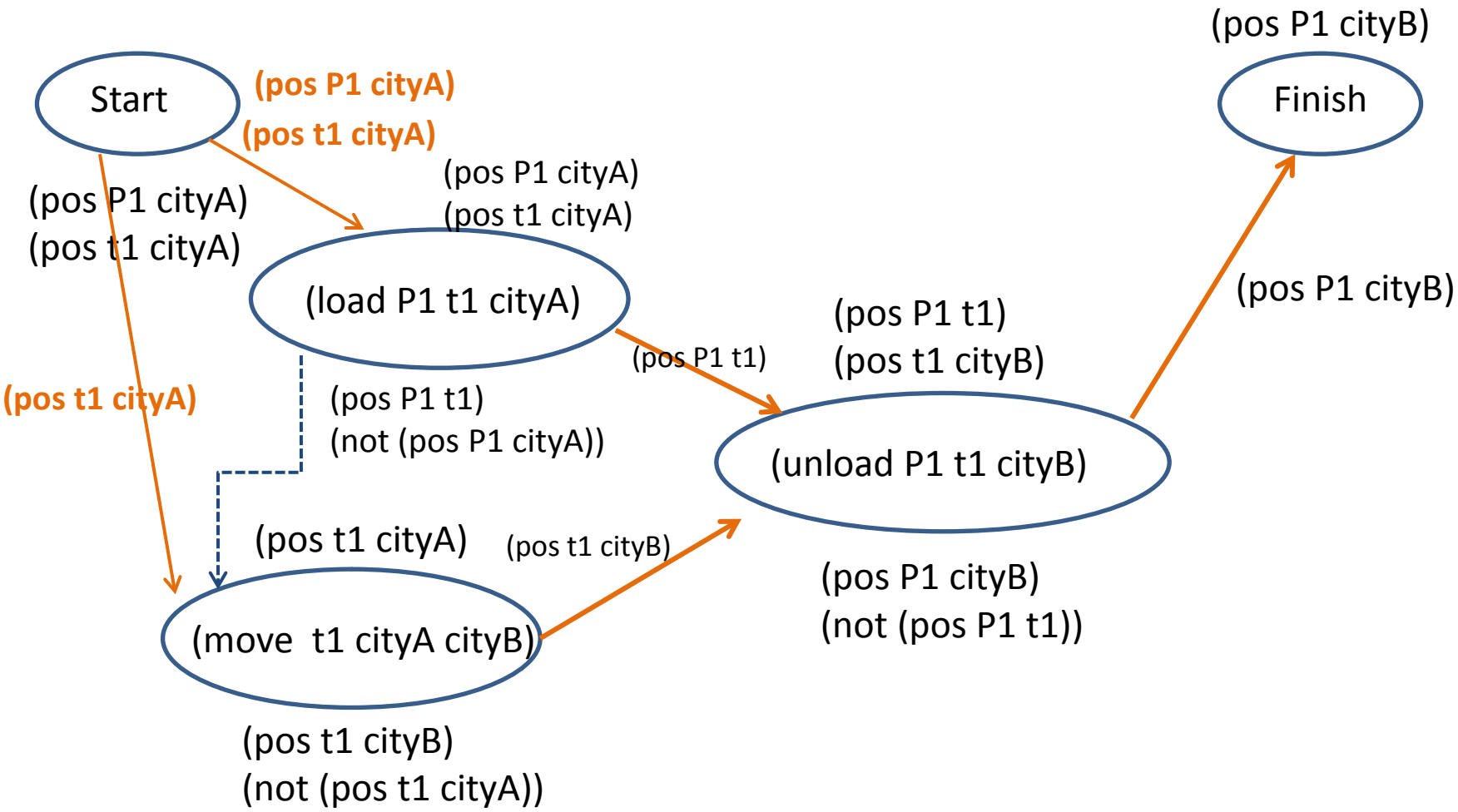
2. State of the art in planning

Planning techniques: partial-order planning (POP)



2. State of the art in planning

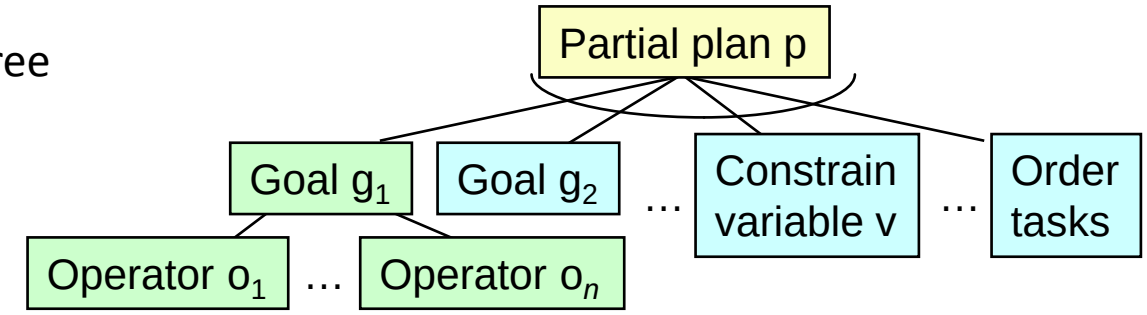
Planning techniques: partial-order planning (POP)



2. State of the art in planning

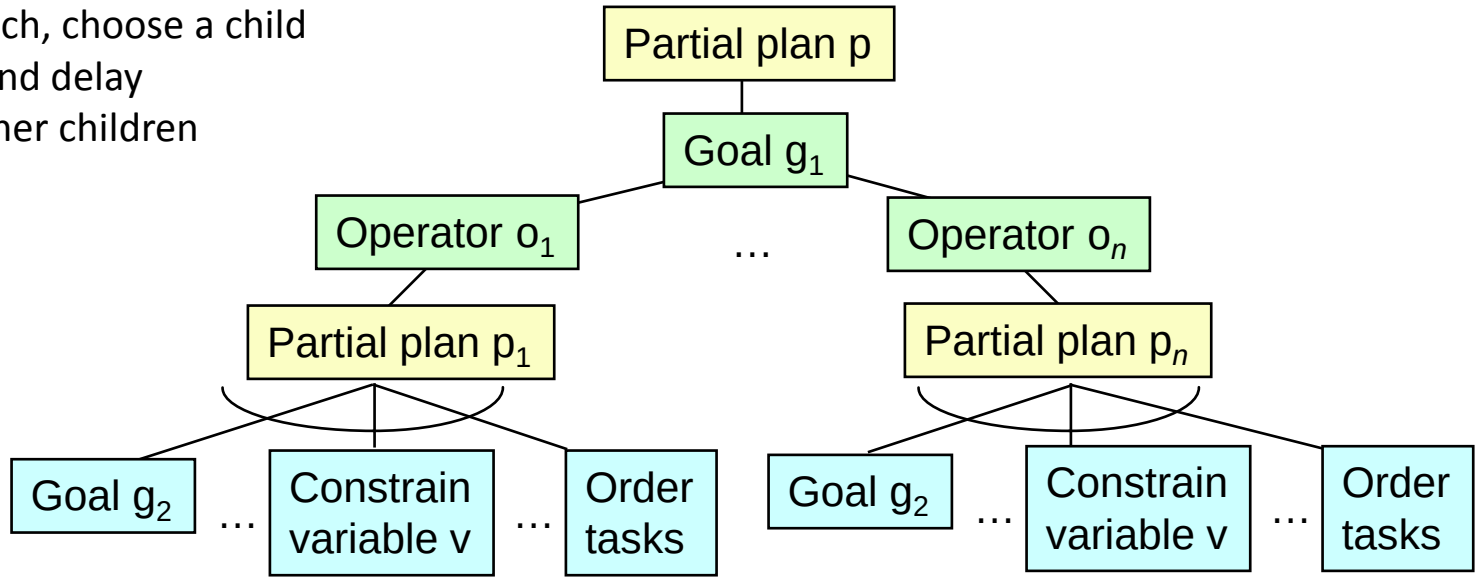
Planning techniques: partial-order planning (POP tree)

The search space is an AND/OR tree



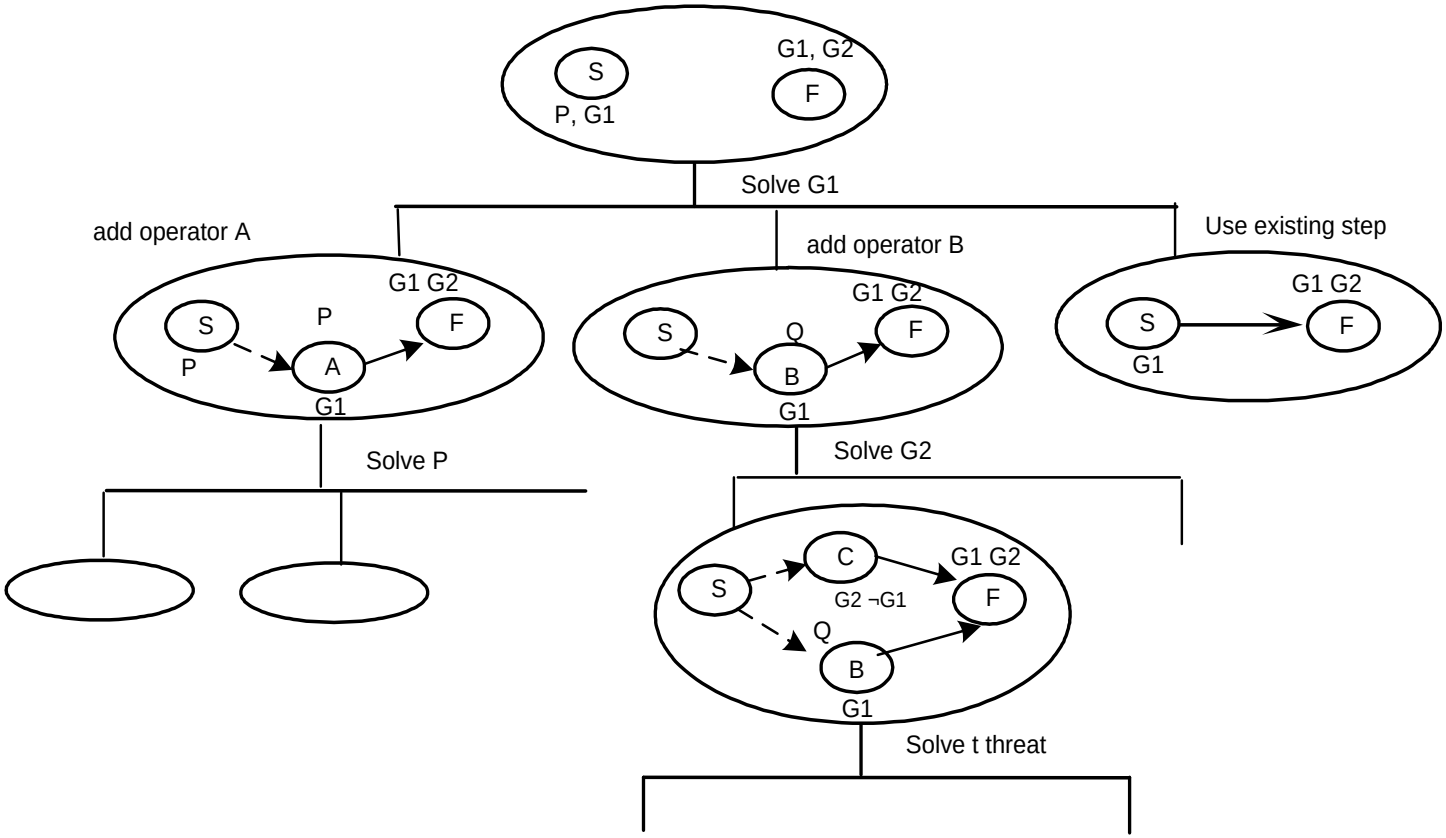
Serializing the tree:
at each AND branch, choose a child to expand next, and delay expanding the other children

POP tree



2. State of the art in planning

Planning techniques: partial-order planning (POP tree)



Heuristics:

- flaw-selection heuristics (ZLIFO)
- resolver-selection heuristics (A*: f(Π)= Steps(Π)+ OpenGoals(Π))

2. State of the art in planning

Planning techniques: planning-graph approach

- TOP and POP -> very large branching factor
- one way to reduce the branching factor → relaxed problem
- Planning-graph approaches rely on the idea of *relaxation of the reachability analysis*
  - Reachability can be computed *exactly* through a *reachability tree* => it cannot be computed in a tractable way
  - Reachability can be *approximated* through a *planning graph* => relaxation of the reachability analysis
- Graphplan: (1) graph expansion (2) solution extraction from the planning graph

procedure Graphplan:

- for k = 0, 1, 2, …

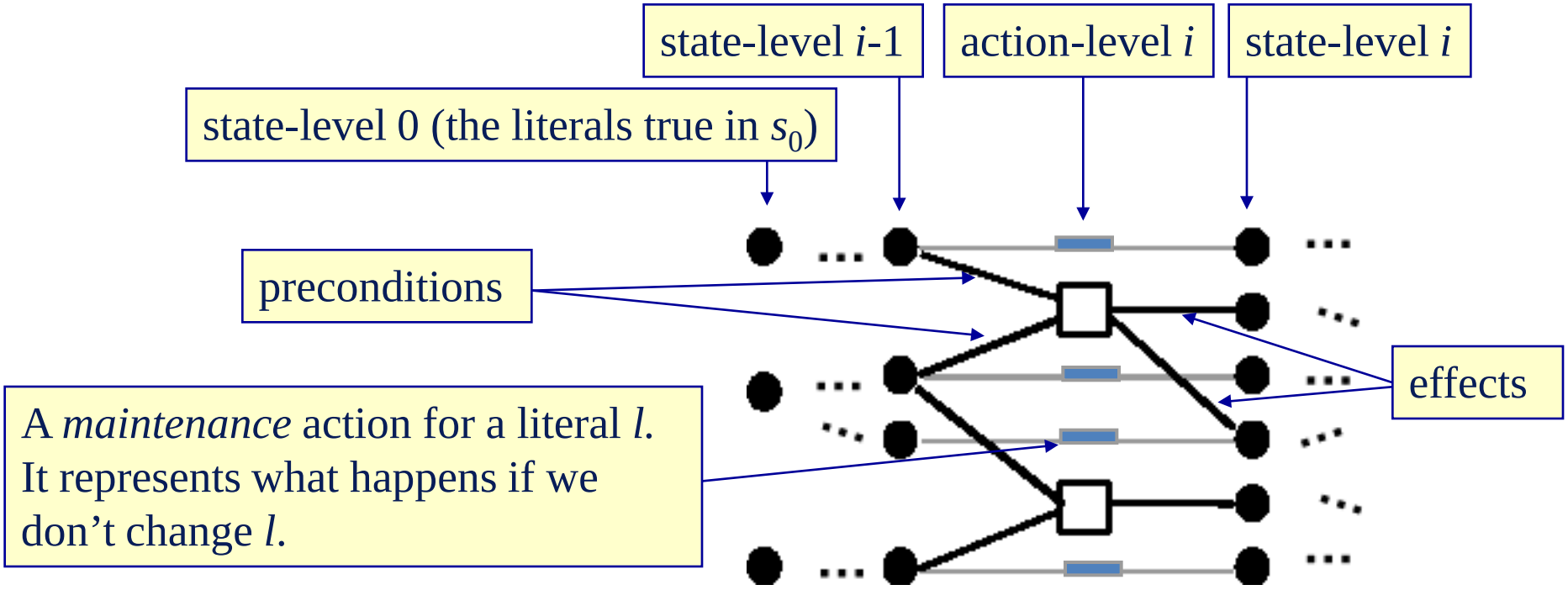
relaxed problem

  - *Graph expansion:*
    - create a "planning graph" that contains k "levels"
  - Check whether the planning graph satisfies a necessary (but insufficient) condition for plan existence
  - If it does, then do *solution extraction*:
    - backward search, modified to consider only the actions in the planning graph
    - if we find a solution, then return it

2. State of the art in planning

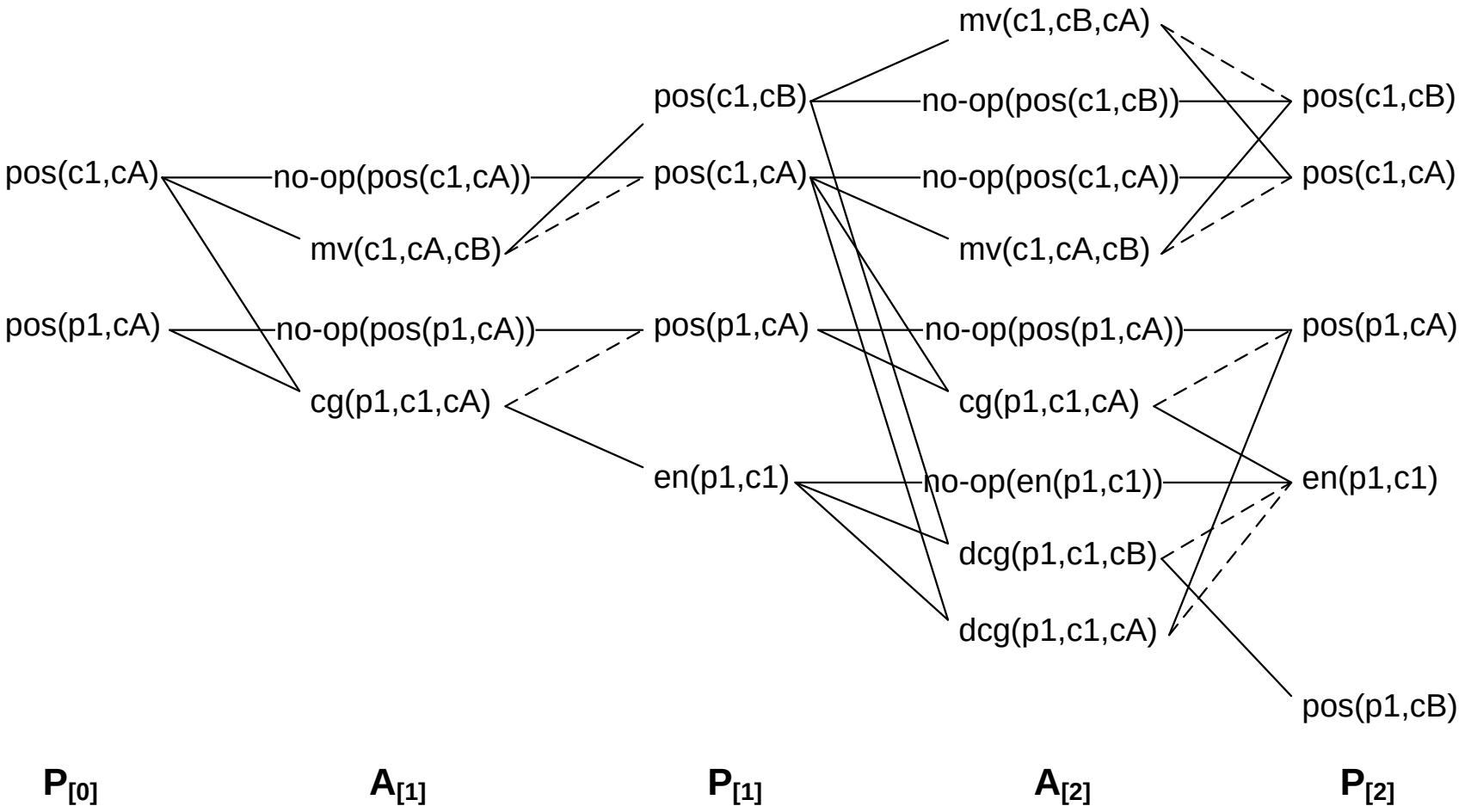
Planning techniques: planning-graph approach

- Search space for a relaxed version of the planning problem
- Alternating layers of ground literals and actions
  - Nodes at action-level *i*: actions that might be possible to execute at time *i*
  - Nodes at state-level *i:* literals that might possibly be true at time *i*
  - Edges: preconditions and effects



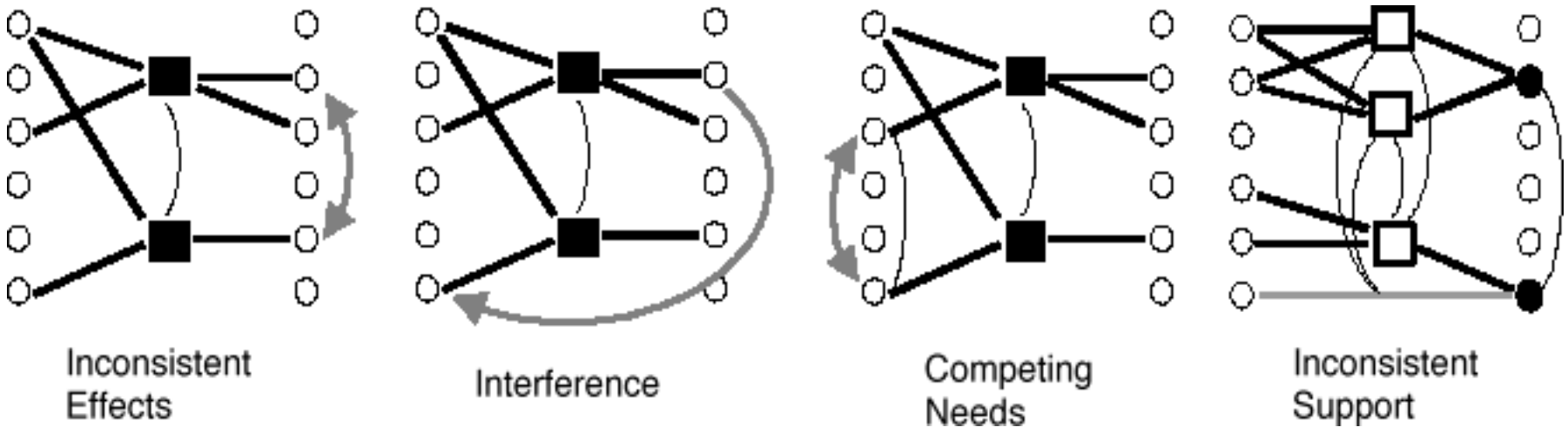
2. State of the art in planning

Planning techniques: planning-graph approach



2. State of the art in planning

Planning techniques: planning-graph approach (Mutual exclusion)



- Two actions at the same action-level are mutex if
  - *Inconsistent effects:* an effect of one negates an effect of the other
  - *Interference:* one deletes a precondition of the other
  - *Competing needs:* **they have mutually exclusive preconditions**
- Otherwise they don't interfere with each other
  - Both may appear in a solution plan
- Two literals at the same state-level are mutex if
  - *Inconsistent support:* one is the negation of the other, **or all ways of achieving them are pairwise mutex**

Recursive propagation of mutexes

2. State of the art in planning

Planning techniques: planning-graph approach (Solution extraction)

Check to see whether there's a possible solution:

(1) All of the goals appear at a proposition level and

(2) None are mutex with each other

The set of goals we are trying to achieve

The level of the state s_j

A real action or a maintenance action

procedure Solution-extraction(*g,j*)

if *j*=0 then return the solution

for each literal *l* in *g*

nondeterministically choose an action to use in state s_{j-1} to achieve *l*

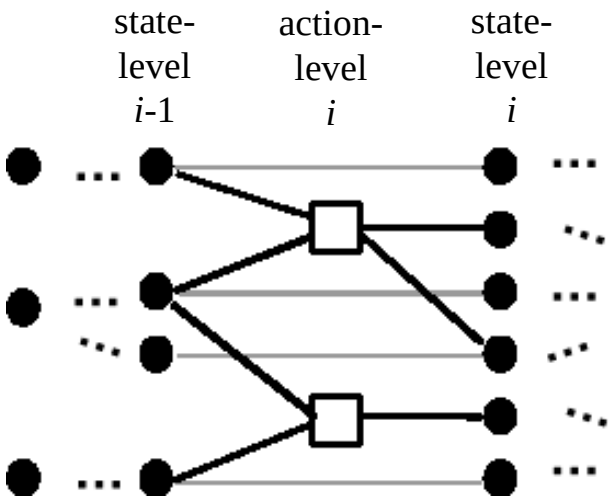
if any pair of chosen actions are mutex

then backtrack

g' := {the preconditions of the chosen actions}

Solution-extraction(*g', j*–1)

end Solution-extraction



2. State of the art in planning

Planning techniques: planning-graph approach (Discussion)

- Advantage:
  - Planning graph in polynomial time
  - The backward-search part of Graphplan—which is the hard part—will only look at the actions in the planning graph
  - smaller search space than POP; thus faster

- Disadvantage:
  - To generate the planning graph, Graphplan creates a huge number of ground atoms (many of them may be irrelevant)
  - The mutual exclusion rules do not guarantee to find *all* mutex relationships, but usually find a large number of them (in fact, determining *all* mutex relationships can be as hard as finding a plan).
  - Mutual exclusion rules only find binary mutex but there also exists other higher-order mutex, eg. *ternary* mutex …

- For classical planning, the advantage outweighs the disadvantage
  - GraphPlan solves classical planning problems much faster than POP

2. State of the art in planning

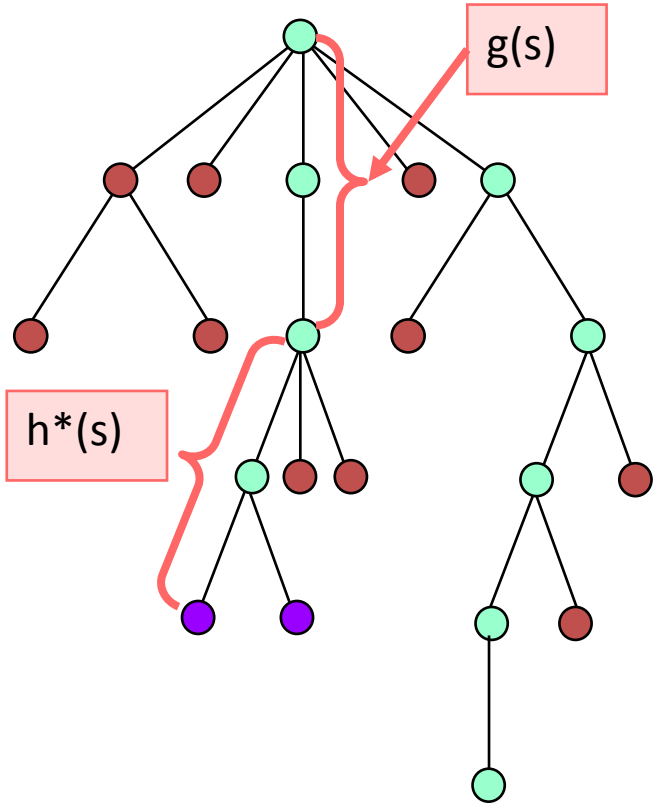
Planning techniques: heuristic planning

- state-based representation, forward search
- node-selection heuristic

- Suppose we're searching a **tree** in which each edge (s,s') has a cost $c(s,s')$
  - If p is a path, let $c(p)$ = sum of the edge costs
  - For classical planning, this is the length of p

- For every state s, let
  - $g(s)$ = cost of the path from s_0 to s
  - $h^*(s)$ = least cost of all paths from s to goal nodes
  - $f^*(s) = g(s) + h^*(s)$ = least cost of all paths from s_0 to goal nodes that go through s

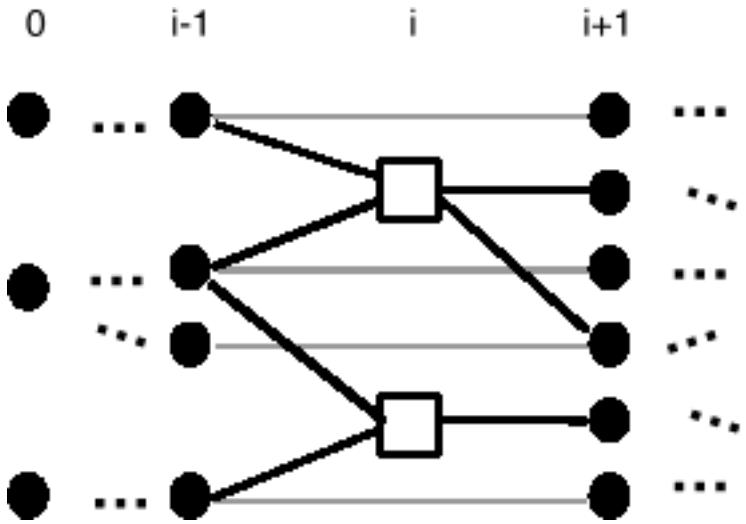
- Suppose $h(s)$ is an estimate of $h^*(s)$
  - Let $f(s) = g(s) + h(s)$
    - $f(s)$ is an estimate of $f^*(s)$
  - h is *admissible* if for every state s, $0 \le h(s) \le h^*(s)$
  - If h is admissible then f is a lower bound on f^*



2. State of the art in planning

Planning techniques: heuristic planning

Heuristics derived from planning graphs



- In the graph, there are alternating layers of ground literals and actions
- The number of “action” layers is a lower bound on the number of actions in the plan
- Construct a planning graph, starting at ***s***
- **h(g_i):** estimate to achieve g_i from **s**
- **h(g_i)=** level of the first layer that “possibly achieves” g_i

2. State of the art in planning

Planning techniques: heuristic planning

Heuristics derived from planning graphs for a set of goals G

- The ***max level heuristic*** takes the maximum level cost of any of the goals (admissible, not very accurate). $hmax(G)=max_{g \in G} h(g)$
- The sum ***level heuristic*** returns the sum of the levelcosts of the goals (inadmissible, works well in practice). $hsum(G)=\sum g \in G\ h(g)$
- The ***max$_2$ level heuristic*** takes the maximum level at which coexist any pair of goals. $hmax_2(G)=_{max\{g1,g2\} \in G} h(g1 \wedge g2)$
- The ***max$_k$ level heuristic*** takes the maximum level at which coexist any *k* goals. $hmax_k(G)=_{max\{g1,g2,\ ..,\ gk\} \in G} h(g1 \wedge g2 \wedge \ \ldots \ \wedge gk)$

2. State of the art in planning

Planning techniques: heuristic planning

Heuristics derived from planning graphs

- Heuristics used by forward state-space planners (like FF)
- Application of the heuristic over each state in the tree
- Planning graph + mutex calculation over each state => very costly process
- Computing heuristics on a **relaxed planning graph:**
  - Ignoring negated effects
  - No mutex calculation
- FF's heuristic: #actions of a plan calculated from the relaxed planning graph

2. State of the art in planning

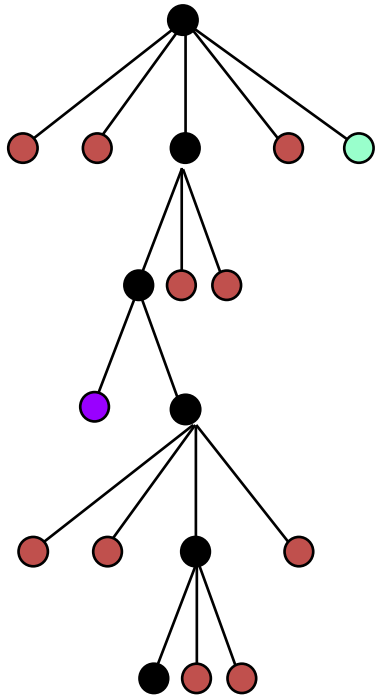
Planning techniques: heuristic planning

FF planner (Jörg Hoffmann, Bernhard Nebel: The FF Planning System: Fast Plan Generation Through Heuristic Search. J. Artif. Intell. Res. (JAIR) 14: 253-302 (2001))

Use a heuristic function $h(s)$ = relaxed plan

Don't want an A*-style search (takes too much memory)

Instead, use a *hill-climbing* procedure:



until we have a solution, do
- expand the current state *s*
- *s* := the child of *s* for which *h(s)* is smallest
  - (i.e., the child we think is closest to a solution)

There are some ways FF improves on this
- e.g. a way to escape from local minima
  - breadth-first search, stopping when a node with lower cost is found

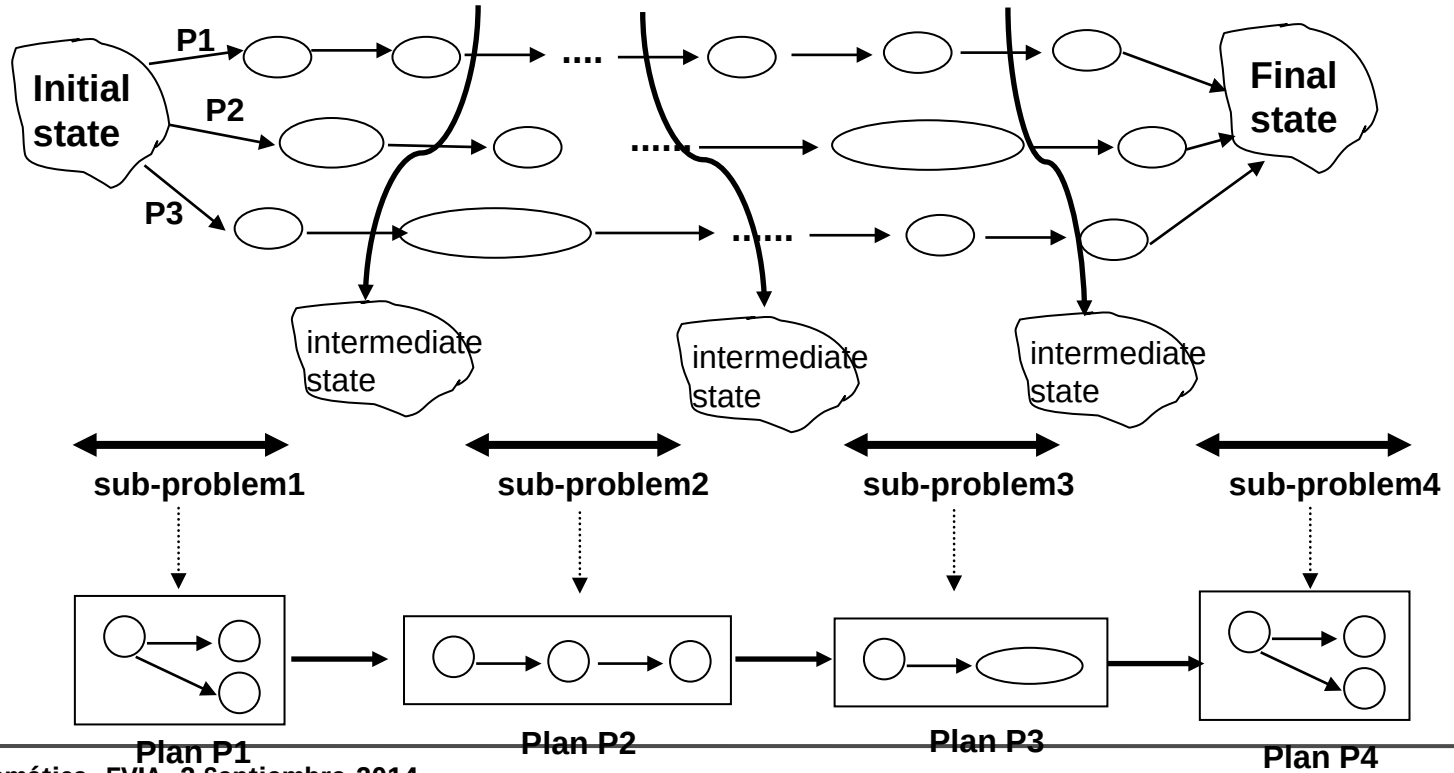
Can't guarantee how fast it will find a solution,
or how good a solution it will find
- However, it works pretty well on many problems

2. State of the art in planning

Planning techniques: decomposition planning

- Traditional planning decomposition:
  - $G = \cup g_i$
  - Concurrent generation of a plan P_i for each goal g_i
  - Solution plan: plan merge to combine the plans P_i

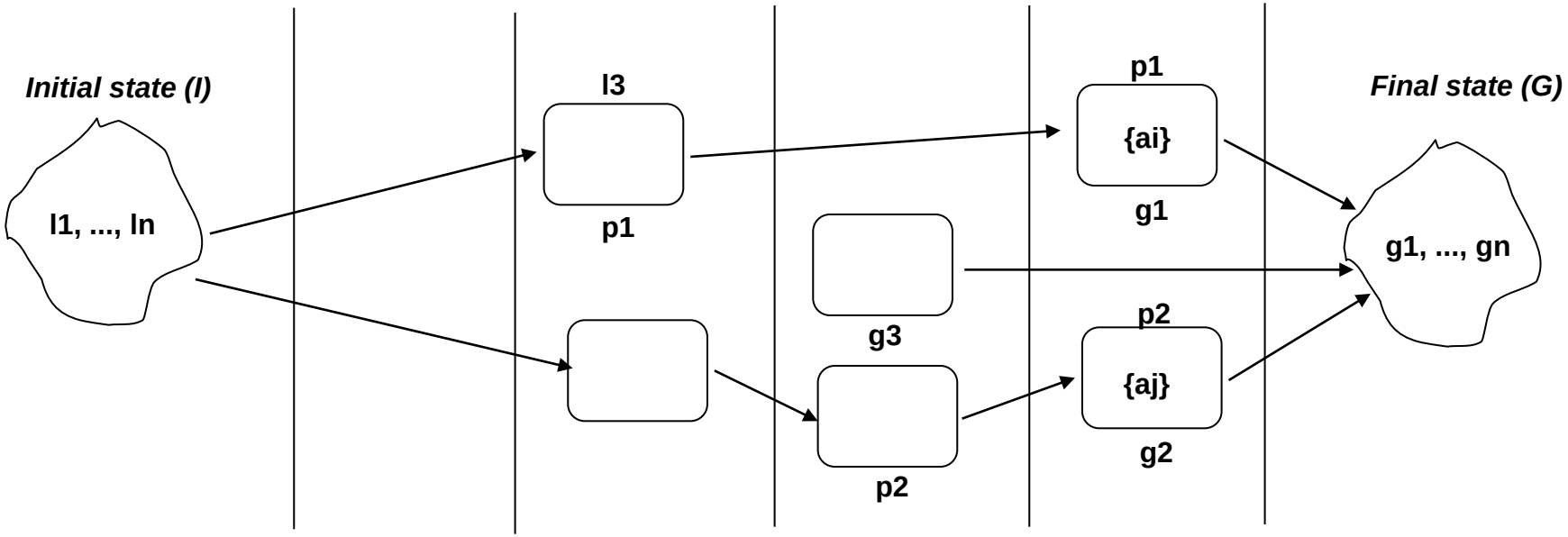
- **A different planning decomposition approach**



2. State of the art in planning

Planning techniques: decomposition planning

- A ***landmark*** is a literal that must be true in ALL solution plans of a planning task (Hoffman, Porteus, Sebastia. *Ordered Landmarks in Planning*. J. Artif. Intell. Res. (JAIR) 22: 215-278 (2004))
- Landmarks of a planning task are extracted through a *Relaxed Planning Graph*

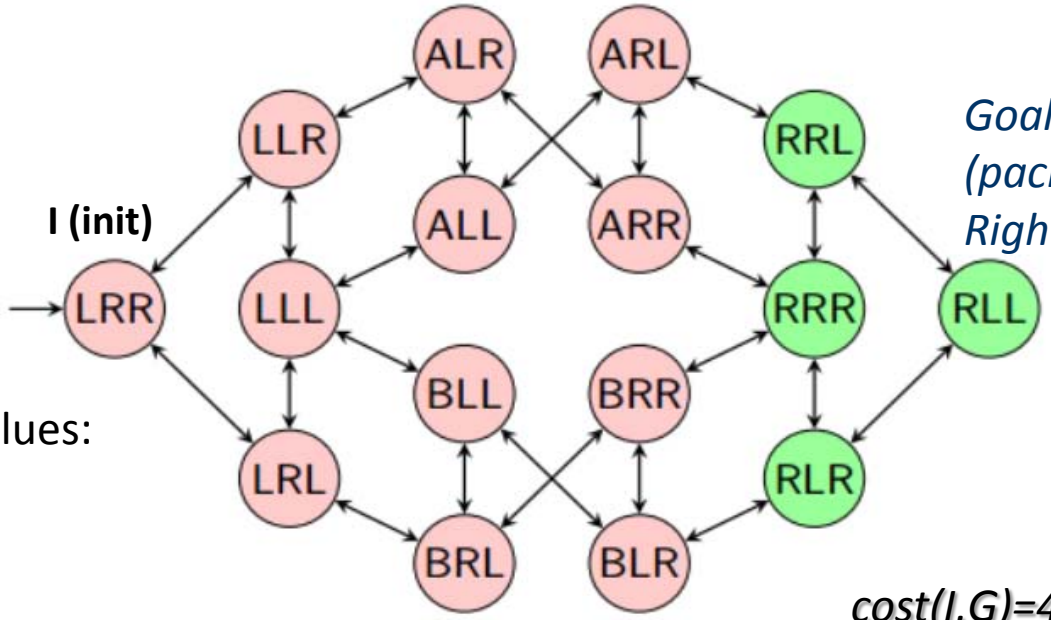


- Landmarks Graph: landmarks along with neccessary/reasonable orders
- *Landmark* is a very relevant concept for solving planning tasks and has been widely exploited in the design of heuristic functions

2. State of the art in planning

Planning techniques: planning abstractions

- Reduce the level of detail, abstract a problem-solving task into higher level representations.
  - **Airplane transportation**: flight schedules on a large network of airports, routes, crew, airport staff, fuel costs, airport local configuration, etc
- Abstraction can reduce branching factor, solution length and likelihood of encountering a deadlock



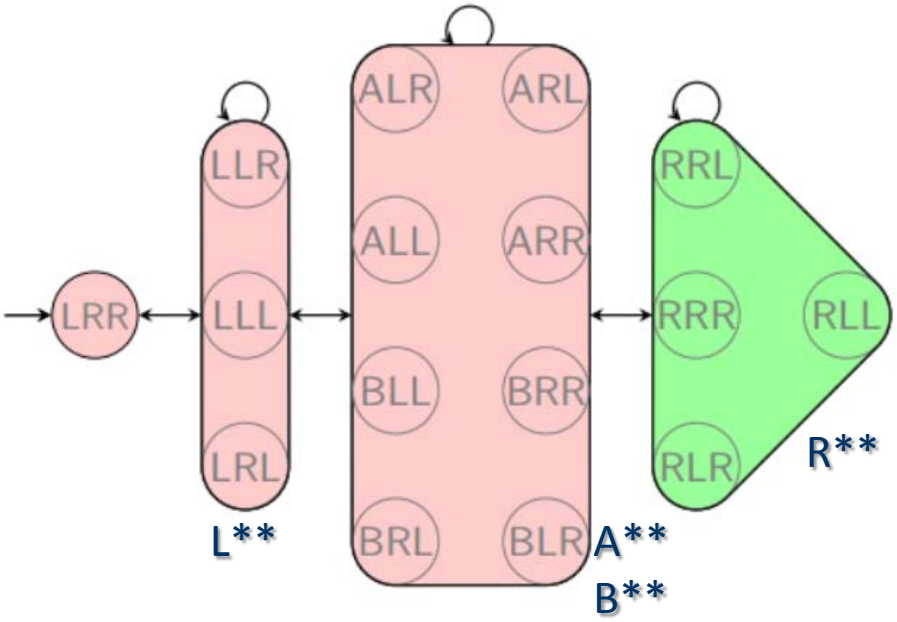
State variables and values:

- (pos package) = L
- (pos truckA) = R
- (pos truckB) = R

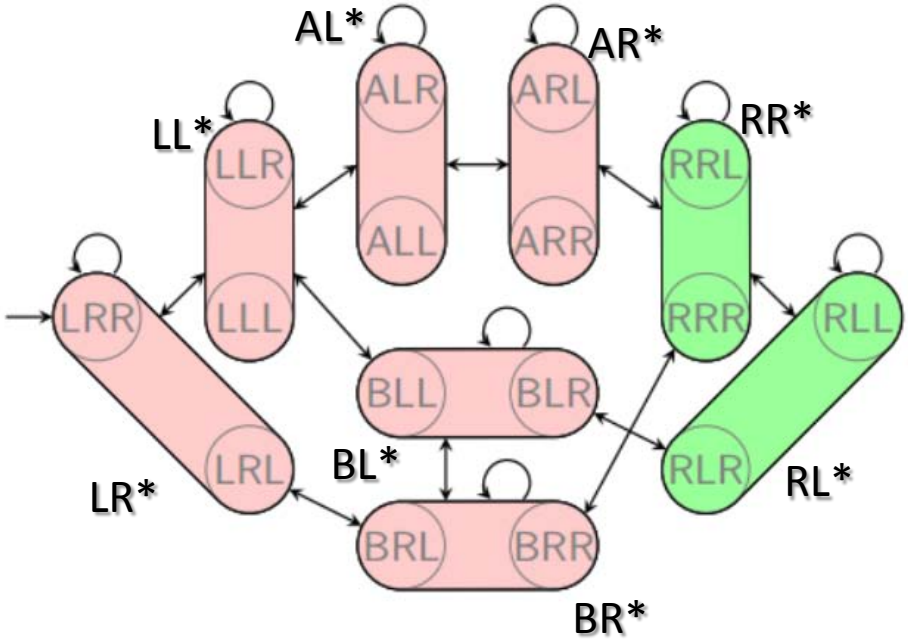
2. State of the art in planning

Planning techniques: planning abstractions

Two possible abstractions:



estimate from I to G = 3



estimate from I to G = 2

2. State of the art in planning

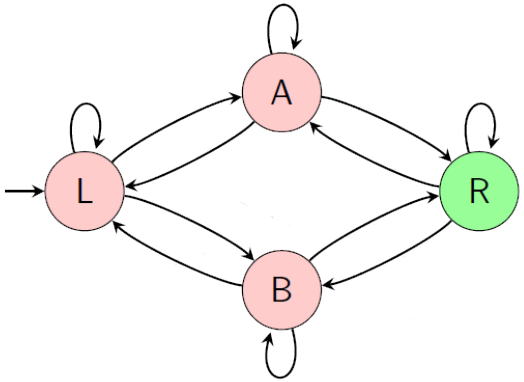
Planning techniques: planning abstractions (merge & shrink)

- Helmert, Haslum, Hoffmann, Nissim: Merge-and-Shrink Abstraction*: A Method for Generating Lower Bounds in Factored State Spaces*. J. ACM 61(3): 16 (2014)
- Abstractions on individual state variables
- Merge the individual projections into a new product abstraction (⊗)
- Due to memory limitations, the product abstractions can be too large . In this case we can shrink them by abstracting them further using any abstraction on an intermediate result, then continue the merging process.

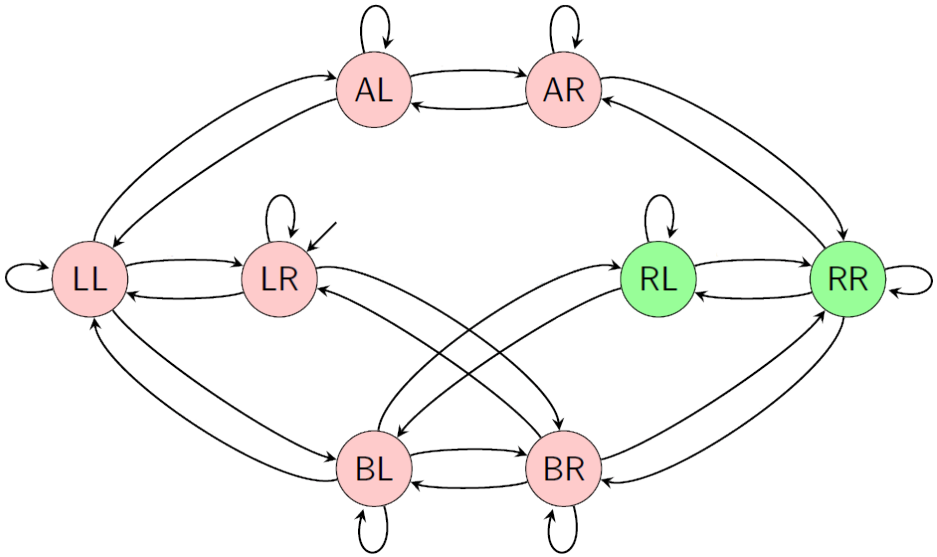
2. State of the art in planning

Planning techniques: planning abstractions (merge & shrink)

abstraction **(pos package)**



abstraction **(pos truckA)**

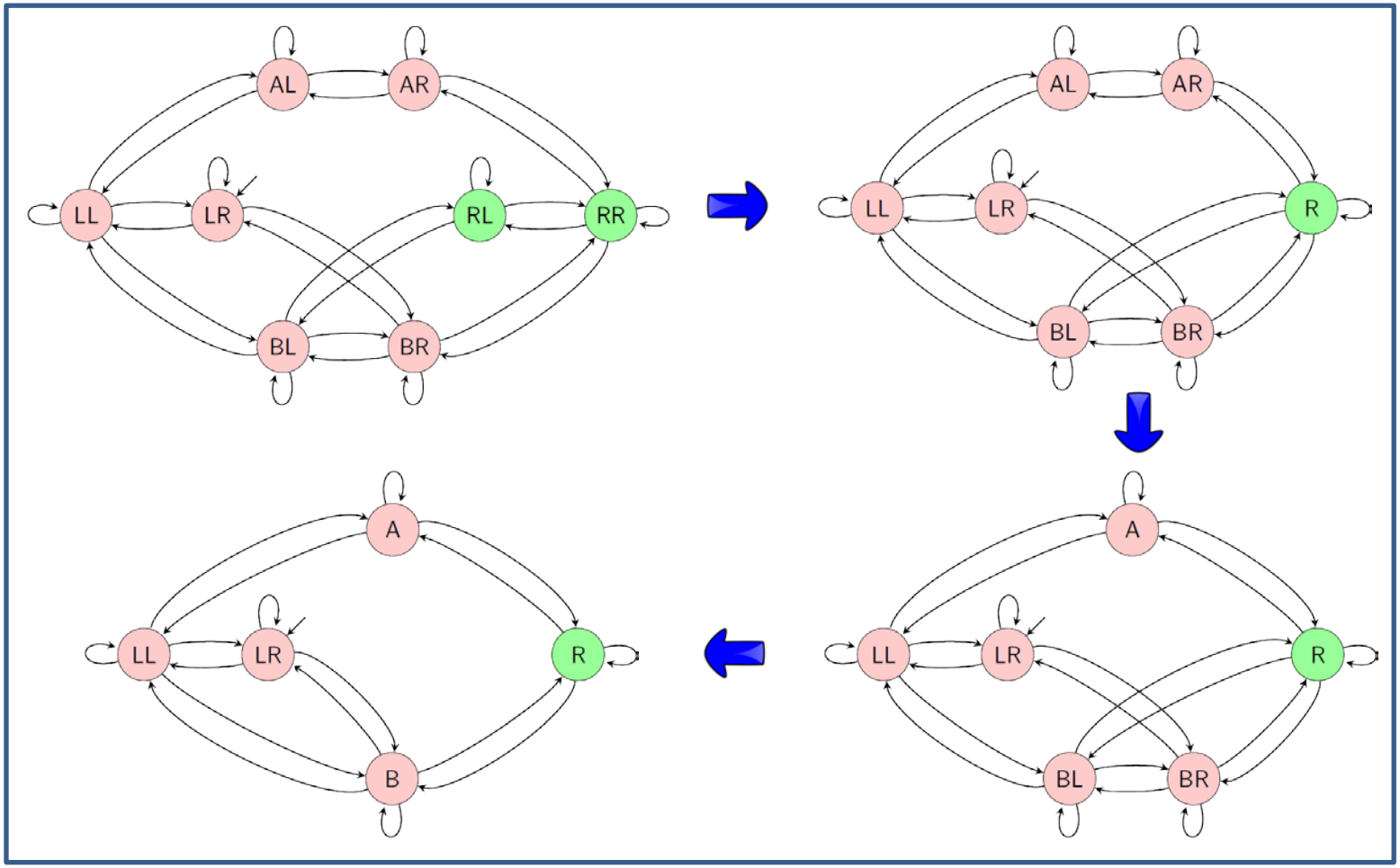


product abstraction **(pos package) ⊗ (pos truckA)**

2. State of the art in planning

Planning techniques: planning abstractions (merge & shrink)

Shrink



Now, on the result of the last shrink we can merge more variables abstractions (pos truckB)

2. State of the art in planning

Planning techniques: Hierarchical Task Network Planning (HTN)

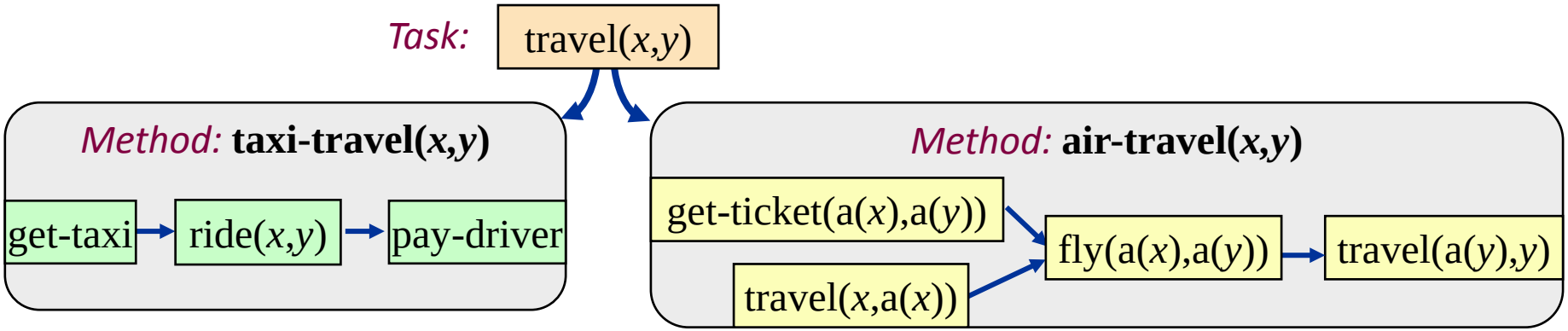
- Action abstraction and plan decomposition hierarchies
- Hierarchical organization of 'actions'
- lowest level reflects directly executable actions (primitive actions)

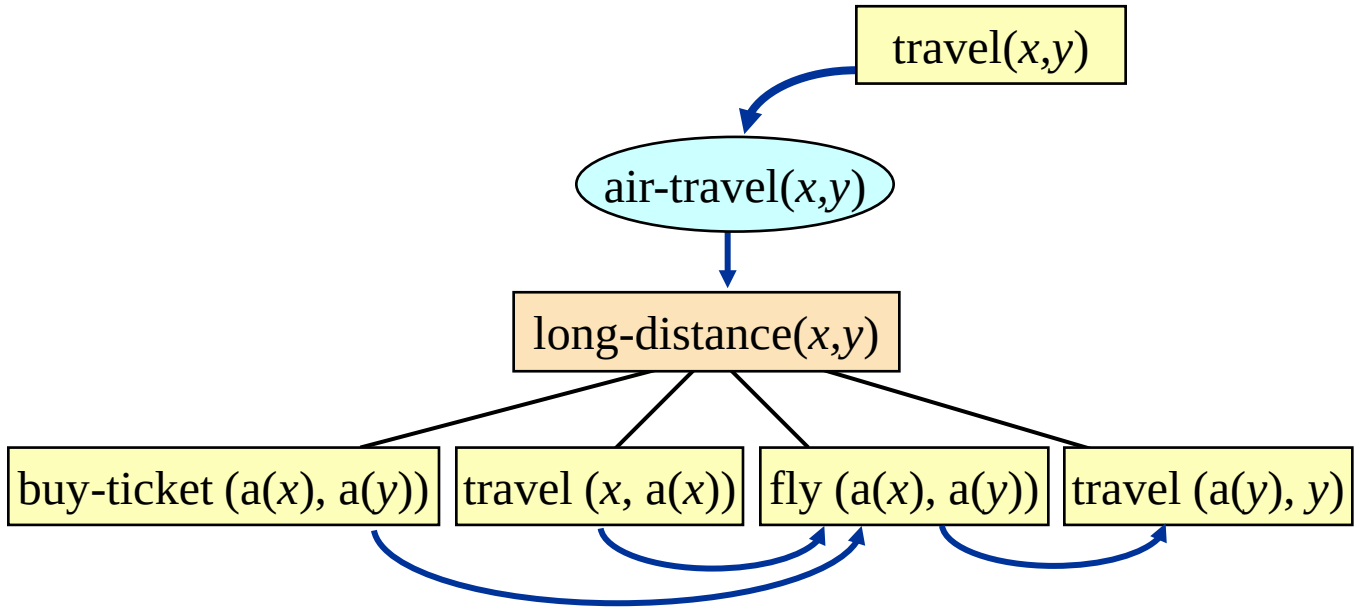
- Planning starts with complex action on top
- Plan constructed through action decomposition
- Substitute complex action with plan of less complex actions

- *Tasks* (activities) rather than goals
- *Methods* to decompose tasks in subtasks
- Enforce constraints
- Backtrack if necessary

2. State of the art in planning

Planning techniques: Hierarchical Task Network Planning (HTN)





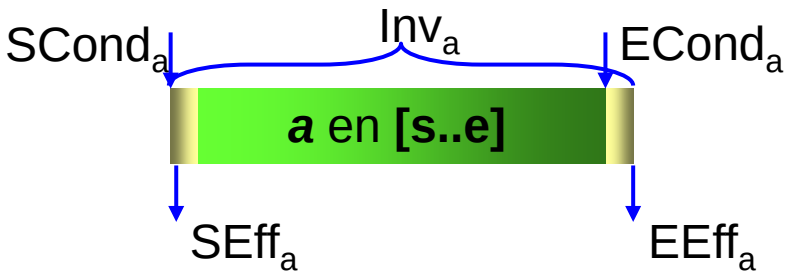
3. Beyond classical planning

- Temporal planning
- Planning with resources (discrete, continuous)
- Planning and Scheduling
- Planning under uncertainty
- Multi-agent planning
- Planning in robotics
- … and more

3. Beyond classical planning

Temporal planning

- Durative actions (actions with duration)
- PDDL2.1 model:

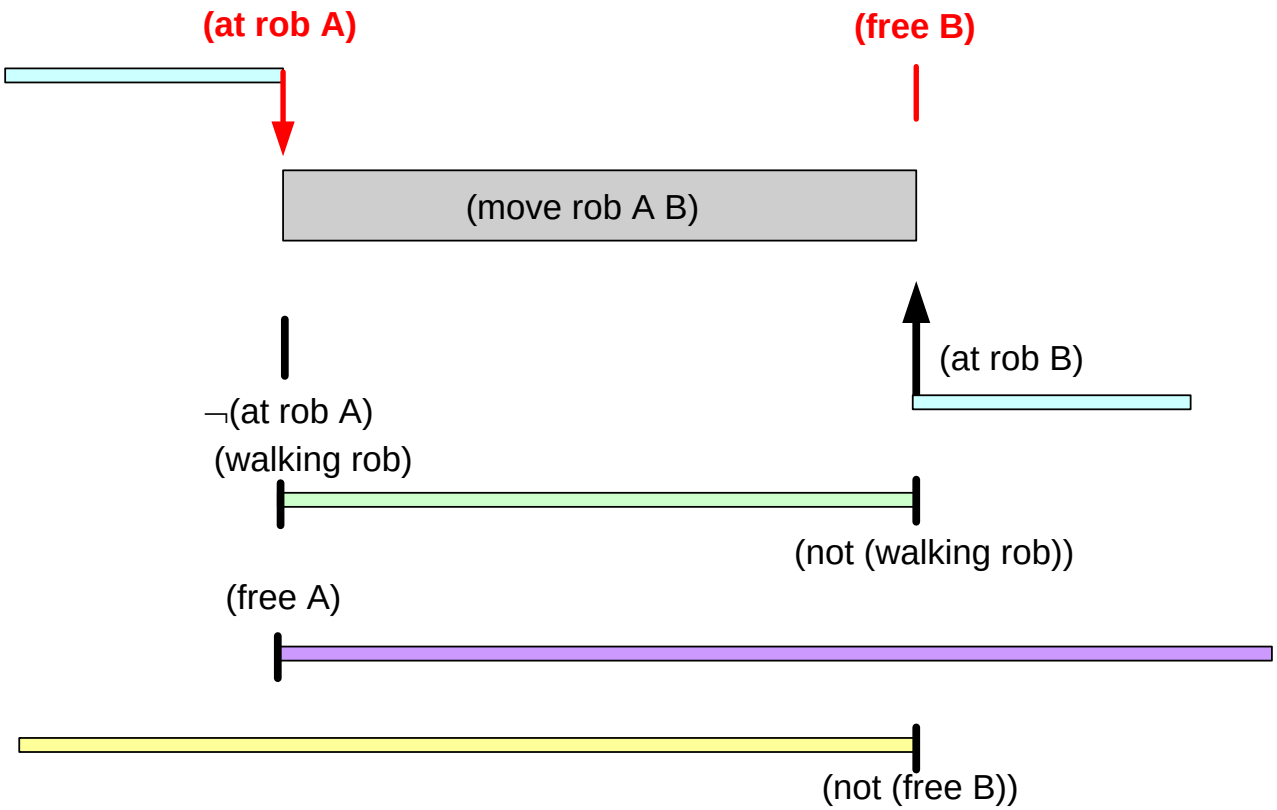



```
(:durative-action board
    :parameters (?p - person ?a - aircraft ?c - city)
    :duration (= ?duration 20)
    :condition (and (at start (at ?p ?c)) (over all (at ?a ?c)))
    :effect (and (at end (in ?p ?a))
                 (at start (not (at ?p ?c))))
```

3. Beyond classical planning

Temporal planning

- Traditional conditions/effects are annotated with the time point of their occurrence
- Multiple choices of concurrency



3. Beyond classical planning

Temporal planning

- PDDL3.0 enables to specify deadlines:

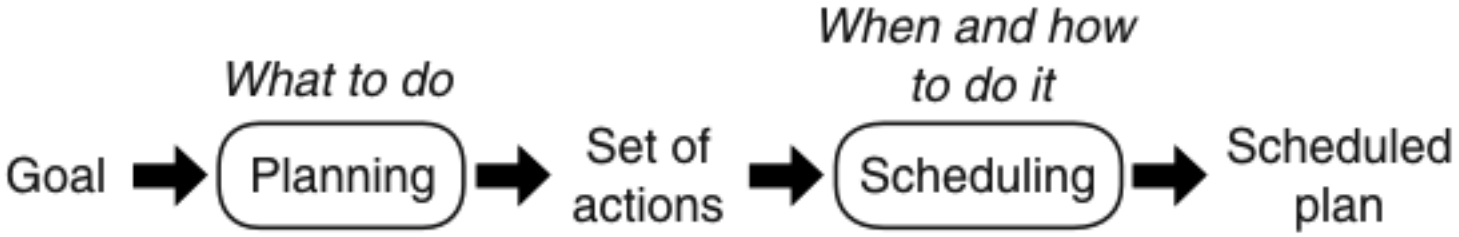
Truck7 must be in cityA before 250 time units	(**within** 250 (at t7 cA))
Whenever the fuel of a truck is below 20, it should be at the refueling post within 10 time units	(forall (?t - truck) (**always-within** 10 (< (fuel ?t) 20) (at ?t fuel-post))))
A truck can visit a certain city only after having visited another particular one	(forall (?t - truck) (**sometime-before** (at ?t c1)(at ?t c2)))

- More expressive temporal models:
  - conditions and effects can be temporally quantified within the interval of execution of the action (planners like ZENO, Sapa, VHPOP, CPT)
  - deadline goals, temporal windows, exogenous events, persistence, quantitative temporal constraints
  - planning solved as a Constraint Satisfaction Problem (CSP)

- Temporal planners:
  - Temporal Graphplan: TPG, TPSYS (PDDL2.1)
  - Action graphs: LPG (PDDL2.1)
  - Heuristic-based: Sapa, TP4
  - POP: HSTS, ZENO, VHPOP (richer models), OPTIC (PDDL3.0), TempLM (PDDL3.0)

3. Beyond classical planning

Planning and Scheduling



resources:

- can be modelled as parameters of an action
  problem: planning algorithms tries out all possibilities (inefficient)
- alternative approach:
  allow unbound resource variables in plan (planning)
  find assignment of resources to actions (scheduling)

resource variables: modified by actions in relative ways
example: move(*r,l,l'*): fuel level changes from *f* to *f-f'*

Let *a* be an action in a planning domain:

attached resource constraints:
required resource: *r*
quantity of resource required: *q*

reusable: resource will be available to other actions after this action is completed

consumable: resource will be consumed when action is complete

3. Beyond classical planning

Planning and Scheduling

- PDDL2.1 enables to handle continuous resources (e.g., energy, benefit, …) as numeric fluents

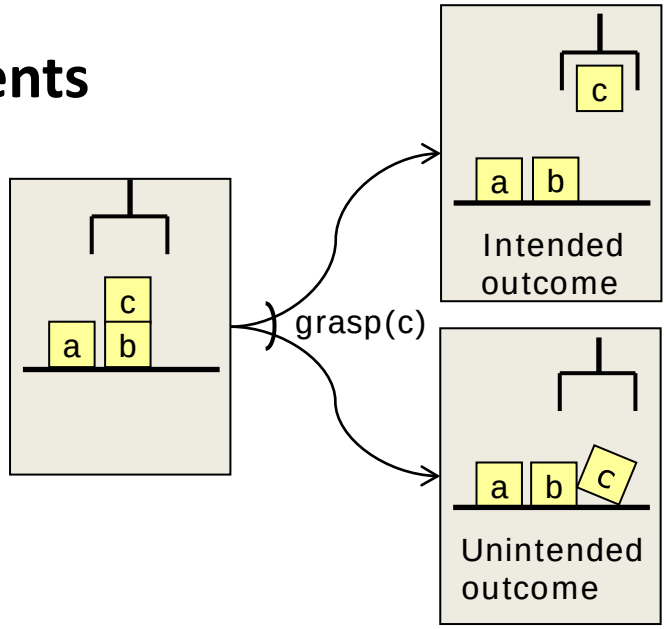
```
(:durative-action fly
 :parameters (?a - aircraft ?c1 ?c2 - city)
 :duration (= ?duration (/ (distance ?c1 ?c2) (slow-speed ?a)))
 :condition (and (at start (at ?a ?c1))
                 (at start (>= (fuel ?a) (* (distance ?c1 ?c2) (slow-burn ?a)))))
 :effect (and (at start (not (at ?a ?c1)))
              (at end (at ?a ?c2))
              (at end (increase total-fuel-used (* (distance ?c1 ?c2) (slow-burn ?a))))
              (at end (decrease (fuel ?a) (* (distance ?c1 ?c2) (slow-burn ?a))))))

(:durative-action refuel
 :parameters (?a - aircraft ?c - city)
 :duration (= ?duration (/ (- (capacity ?a) (fuel ?a)) (refuel-rate ?a)))
 :condition (and (at start (> (capacity ?a) (fuel ?a)))
                 (over all (at ?a ?c)))
 :effect (at end (assign (fuel ?a) (capacity ?a))))
```

- Planning with numeric variables and multi-objective planning (minimize distance, fuel, total time, etc.
- Planning techniques based on Relaxed Planning Graph: SAPA, SimPlanner, TPSYS, Metric-FF

3. Beyond classical planning

Planning in non-deterministic environments

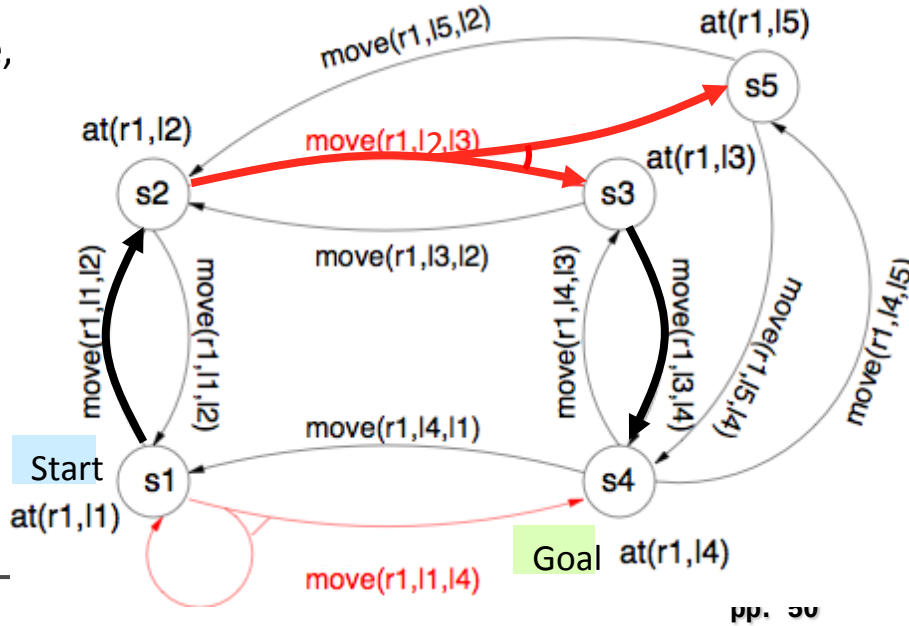
- Non-determinism:
  - Uncertainty in the action effects, multiple possible outcomes (action failures, exogenous events)
  - Uncertainty in the initial state (the current state is in general not known)



- *Nondeterministic systems* are like Markov Decision Processes (MDPs), but without probabilities attached to the outcomes
  - Useful if accurate probabilities aren't available, probability calculations would introduce inaccuracies

- *Nondeterministic system*: a triple $\Sigma = (S, A, \gamma)$
  S = finite set of states
  A = finite set of actions
  $\gamma: S \times A \rightarrow 2^S$

We can also have *nondeterministic policies :* multiple actions for some states ($\pi: S \rightarrow 2^A$)



3. Beyond classical planning

Planning in non-deterministic environments

- Partial observability:
  - The state of the system is only partially visible at run-time
  - Different states of the system are indistinguishable for the controller
  - Some variables may never be observable by the controller (e.g., microprocessor)
  - Some variables can be observable in some states or only after some 'sensing actions' (e.g., robot ignores if door is open in another room, web service composition)
  - Observations returns sets of states rather than single states (power set)

- Extended goals:
  - Goals need to specify requirements that take into account nondeterminism and possible failures
  - For instance, guarantee that some safe state is maintained
    - the robot must avoid dangerous places
    - the tank temperature must never be above 40ºC
  - Extended goal: complex goal involving temporal conditions, conditions to be maintained rather than reached

3. Beyond classical planning

Planning in non-deterministic environments

	Planning based on model checking	Planning based on Markov Decision Processes (MDP)
Characteristics	non-determinism partial observability extended goals	non-determinism partial observability extended goals probabilities
Planning domain	non-deterministic state-transition system	stochastic system
Extended goals	formulas in temporal logic* (*conditions* on the entire plan execution)	utility functions (e.g., costs and rewards) (*preferences* on the entire plan execution)
Plans	conditional plans	policies
Planning	control the evolution of the system (check temporal formulas)	optimization problem (maximize the utility function)
Techniques	symbolic model checking techniques that use Ordered Binary Decision Diagrams (OBDD)	probability distribution over the state space (belief states)

* Formulas in temporal logic are also used to express reachability goals

3. Beyond classical planning

Multi-agent Planning (MAP)

- MAP is a social approach to planning by which multiple intelligent entities work together to solve planning tasks that they are not able to solve by themselves, or to at least accomplish them better by cooperating

- Distribution of information:
  - Spatially: agents have different knowledge of the problem
  - Functionally: agents have different capabilities
  - Privacy

- Cooperative MAP task: $T_{MAP}=\{AG, I, A, G\}$
  - AG: set of agents
  - I: initial state of the planning task ($\cup I^i$)
  - A: set of actions of the agents ($\cup A^i$)
  - G: set of goals of T_{MAP}; G is common to all agents in AG, that is, all of the agents are aimed at solving G
  - Classical planning techniques extended to multi-agent context

- Planning for self-interested agents:
  - Agents have different goals or preferences (soft goals)
  - Agents execute their plans in a common environment ➔ conflicts
  - Techniques: social choice and game-theory

Planificación Automática

Escuela de Verano de Inteligencia Artificial

Eva Onaindía

3 Septiembre 2014