Introduction to AI

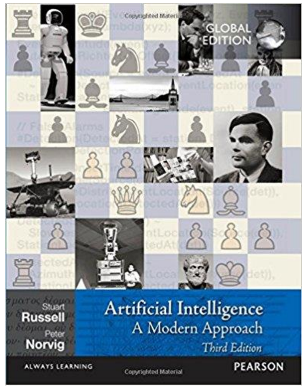
-Planning-

Francesca Toni

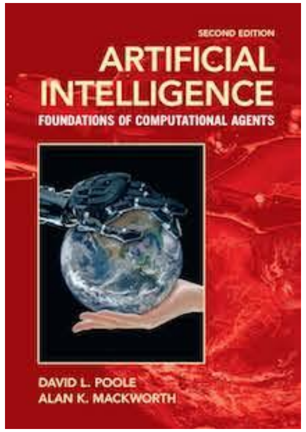
Outline

- Classical planning
- Planning as search
- Planning algorithms: graph-plan

Recommended reading:
(most of) Chapter 10

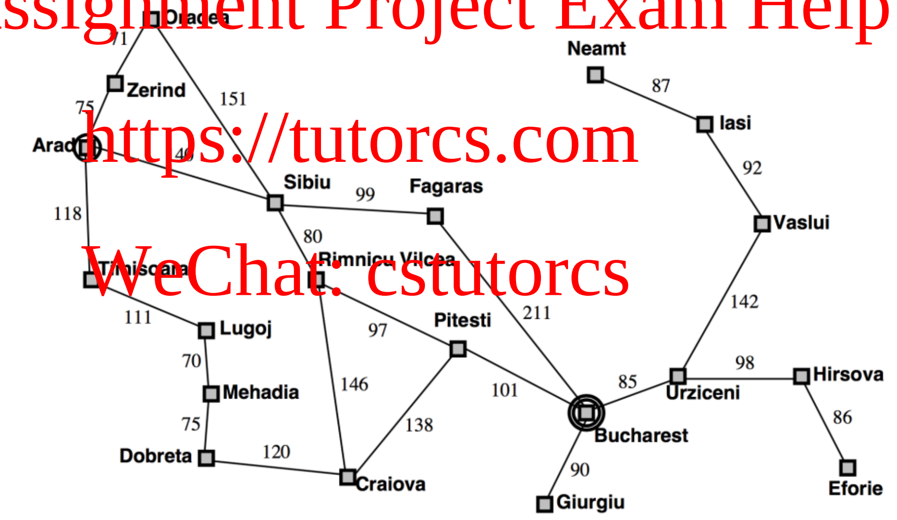


Additional reading:
Chapter 6



(Classical) planning

- Intelligent agents have goals
- Plans are sequences of actions allowing to achieve those goals, from a given initial state



- Planning amounts to selecting (optimal) plans
- Plans then need to be executed in the world

Which problems can be solved by classical planning?

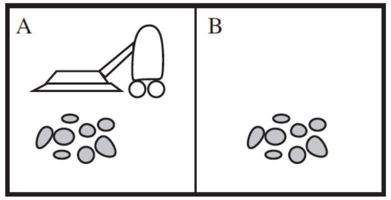
Environment is

- Observable
  - Current state known
- Discrete
  - From each state finitely many next states by executing actions
- Deterministic
  - Each action has exactly one outcome (next state)
- Known
  - Possible actions and next states for each state

Classical planning = Search?

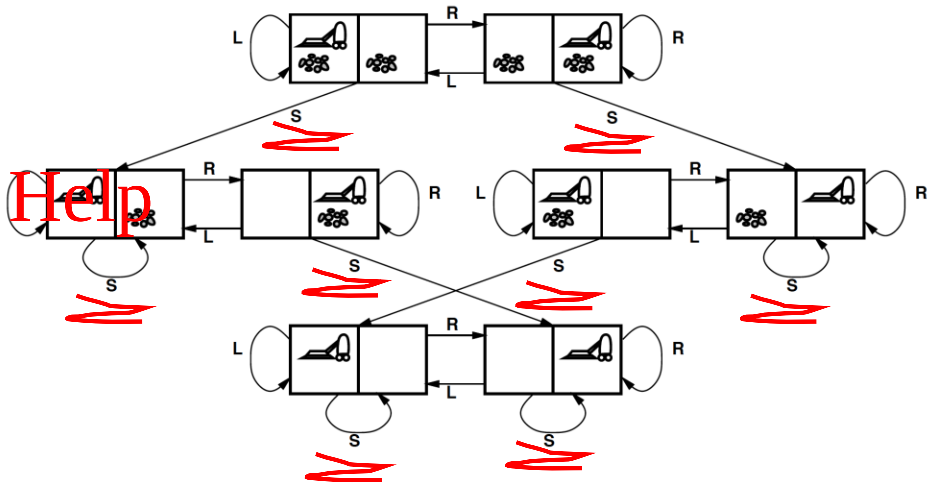
Representations for planning: vacuum world example

Start:



Goal: no dirt

Search Graph:



Start: $In(A) \wedge Dirt(A) \wedge Dirt(B)$
Goal: $\neg Dirt(A) \wedge \neg Dirt(B)$

Action descriptions:
action($S(x)$), precondition($S(x)$,$In(x)$), postcondition($S(x)$,$\neg Dirt(x)$)

$In(x)$

$\boldsymbol{S(x)}$

$\neg Dirt(x)$

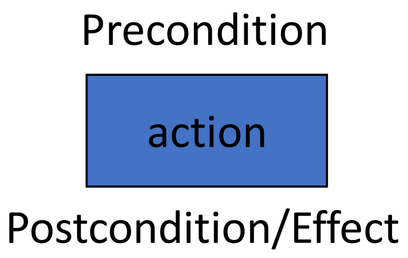
STRIPS (STanford Research Institute Problem Solver)

Special purpose, restricted language for planning

*atoms in the original STRIPS

- **States** are conjunctions of (function-free) ground atoms (positive *fluents*)
- **Goals** are conjunctions of (function-free) literals* (positive or negative *fluents* - possibly containing variables, *implicitly* existentially quantified)

- **Operators** (schemata – may contain variables, implicitly universally quantified):
  - Action description = name of action
  - Precondition = conjunction of literals*
  - Postcondition (Effect) = conjunction of literals
  - Every variable in effect must appear in precondition



- **Actions** are fully instantiated operators: different instances of the same operator are different actions

STRIPS – syntactic assumptions

- $In(x)$ cannot be part of a state (as non-ground)
- $\neg Dirt(A)$ cannot be part of a state (as not an atom)
- $In(Adjacent(x))$ cannot be part of a state (as $Adjacent$ is a function)

- $\forall x\ \neg Dirt(x)$ cannot be a goal (as universally quantified)
- $\neg\, Dirt(x)$ can be a goal (implicitly amounting to $\exists x\ \neg Dirt(x)$)

$P(x)$

- $\boxed{\boldsymbol{A(x)}}$ cannot be an action schema (as y does not occur in $P(x)$)

$Q\,(x,y)$

STRIPS – “semantic” assumptions

- Fluents not “mentioned” in a state representation are (implicitly) false

$In(A) \land Dirt(A) \land Dirt(B)$: implicitly $\neg In(B)$

- Fluents persist (do not change) unless they are explicitly changed by an action

$In(x)$

$\boxed{\boldsymbol{S(x)}}$

$\neg Dirt(x)$

$$In(A) \land Dirt(A) \land Dirt(B) \xrightarrow{S(A)} In(A) \land Dirt(B)$$

- Time is implicit

$In(x)$

$\boxed{\boldsymbol{S(x)}}$ Time t

$\neg Dirt(x)$ Time t+1

Execution of an action

$In(x)$

$\boxed{\boldsymbol{S(x)}}$

$\neg Dirt(x)$

- The action needs to be applicable in the current state:

  the state entails the preconditions of the action

$S(A)$ is applicable in current state $In(A) \land Dirt(A) \land Dirt(B)$

- The result is a new state, modifying the current state "using the fluents in the effects":

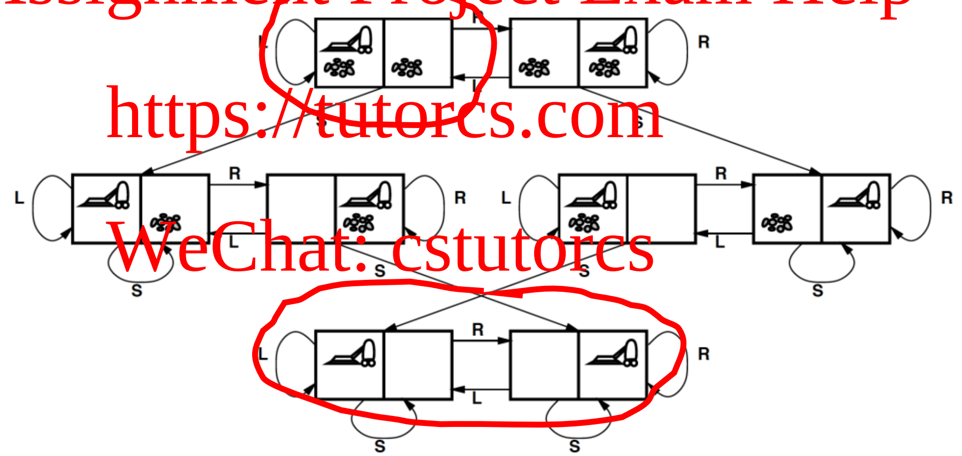
  new state= current state – negation of negative fluents in effects

  + positive fluents in effects

executing $S(A)$ in current state removes $Dirt(A)$ giving $In(A) \land Dirt(B)$

Solving a planning problem

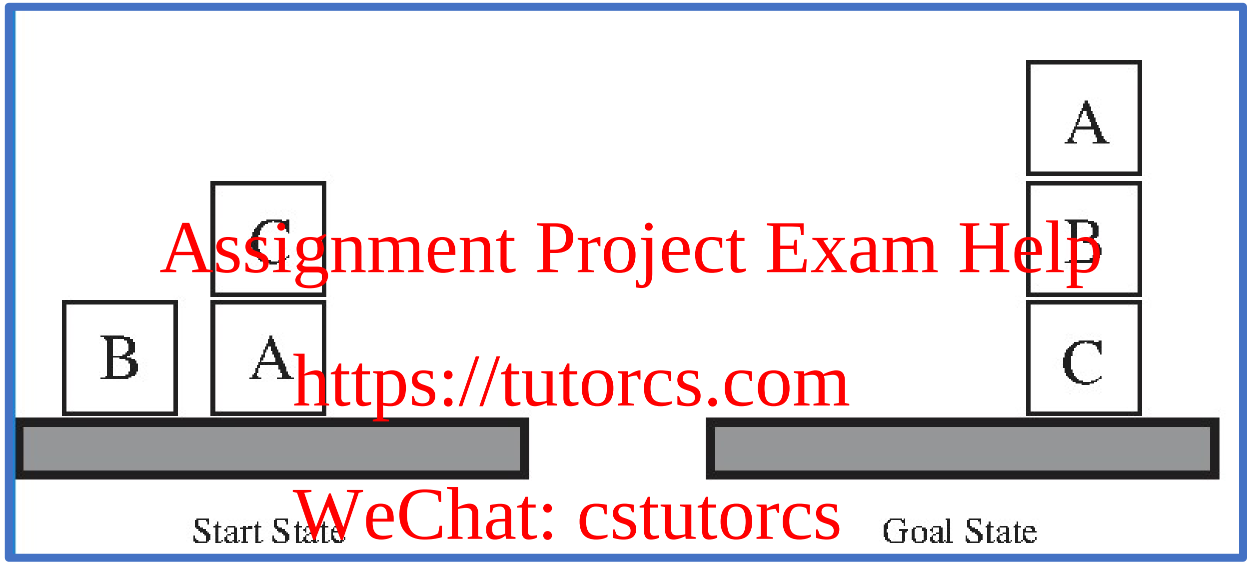
Find a sequence of actions from the initial state to a state in which the goal holds (i.e. entailing the goal)



Start: $In(A) \land Dirt(A) \land Dirt(B)$
Goal: $\neg Dirt(A) \land \neg Dirt(B)$

A plan: S(.), R(.), S(.) **Another plan: R(.),S(.),L(.),S(.)**

The blocks' world



$On(b,x), Clear(b), Clear(y), b \neq x \neq y, b \neq Table, y \neq Table$

$\boxed{Move(b,x,y)}$

$On(b,y), Clear(x), \neg On(b,x), \neg Clear(y)$

$On(b,x), Clear(b), b \neq x \neq Table$

$\boxed{Move2Table(b,x)}$

$On(b,Table), Clear(x), \neg On(b,x)$

A plan: $Move2Table(C,A), Move(B,Table,C), Move(A,Table,B)$

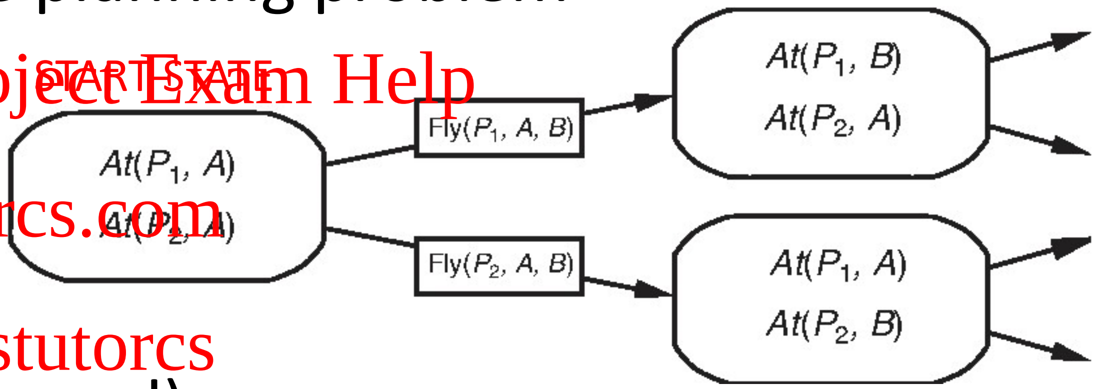
Planning as search

- Progression planning (forward)
  - (blind or heuristic) search with the **graph** obtained from the specification of the planning problem

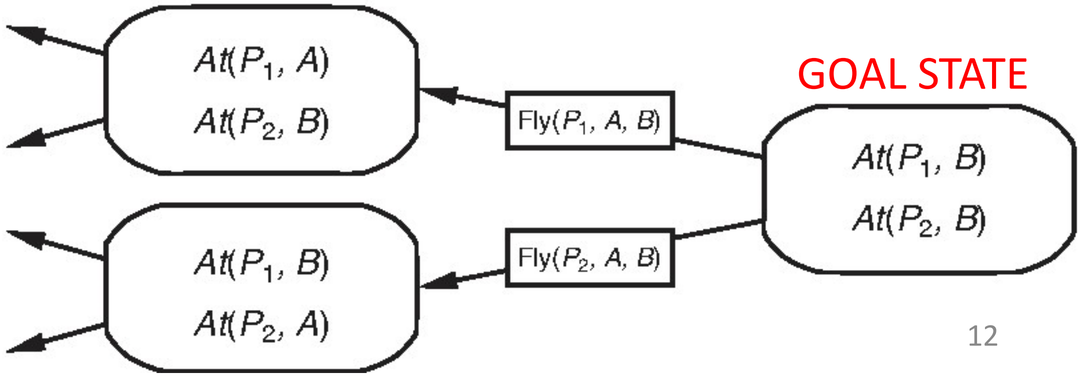
$At(p,x), Plane(p), Airport(x)$

$Fly(p,x,y)$

$At(p,y), \neg At(p,x)$



- Regression planning (backward)
  - (blind or heuristic) search backwards from the goal (using predecessors of actions)



Progression planning needs good heuristics

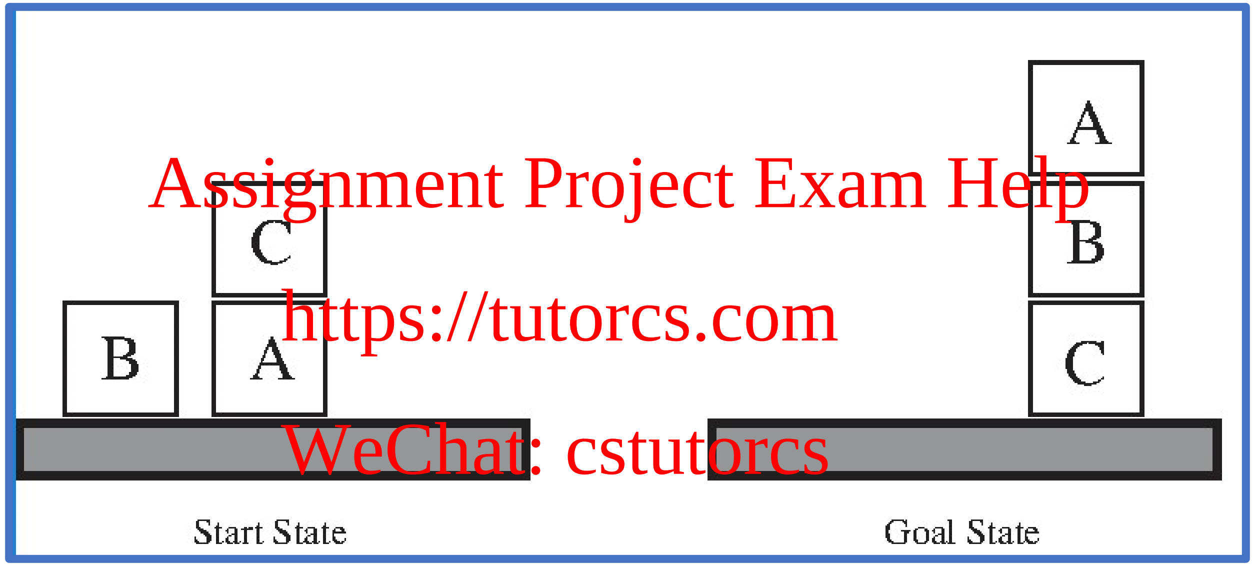
Planning problems tend to have

- large search spaces, with
- many (irrelevant) actions

Good heuristics from "relaxed" problems, e.g.

1. add edges (e.g. by ignoring preconditions and effects not in goal)

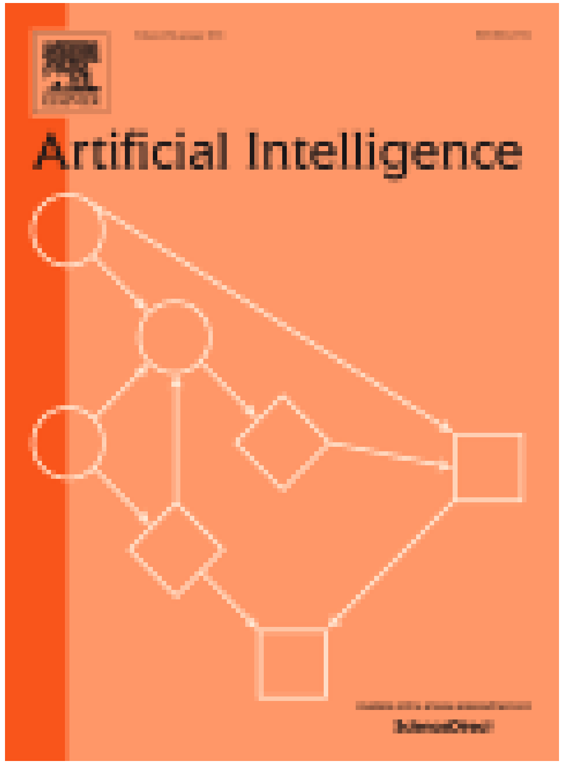
Non-interleaved regression planning is incomplete (Sussman anomaly)



Non-interleaved (regression) planner: 2 subgoals $On(B, C), On(A, B)$

- Either start with: $Move(B, Table, C)$
- Or start with: $Move2Table(C, A), Move(A, Table, B)$

graph-plan algorithm



2017 CLASSIC PAPER AWARD

Fast Planning Through Planning Graph Analysis.
Avrim Blum, Merrick L. Furst
Artif. Intell. 90(1-2): 281-300 (1997)

This seminal paper changed the perspective on classical planning algorithms.

Before the paper appeared, most of the planning approaches used back-chaining methods searching in plan space. Blum and Furst instead proposed to create a particular graph structure in an iterative deepening fashion for constraining
a backward search from the goal, leading to a dramatic performance increase. Although the specific planning algorithm proposed by the authors did not prevail, the ideas behind the algorithm and the empirical methodology adopted, inspired
current approaches such as SAT-based planning and heuristic search-based planning methods. The work also demonstrated that it is quite worthwhile to go off the beaten path and take a fresh view on existing algorithmic problems.

Planning graph

$Have(Cake)$

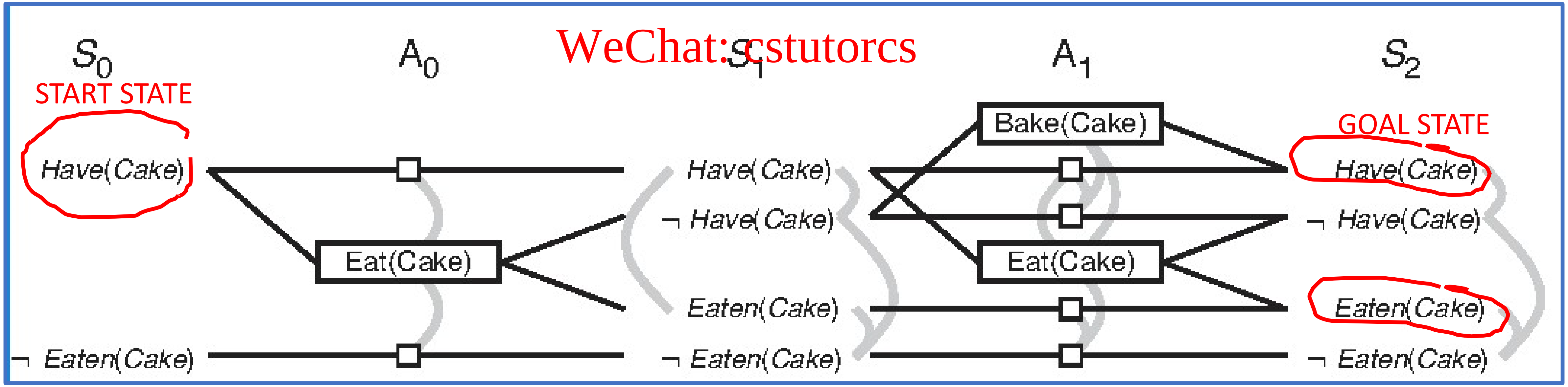
$Eat(cake)$

$\neg Have(Cake), Eaten(Cake)$

$\neg Have(Cake)$

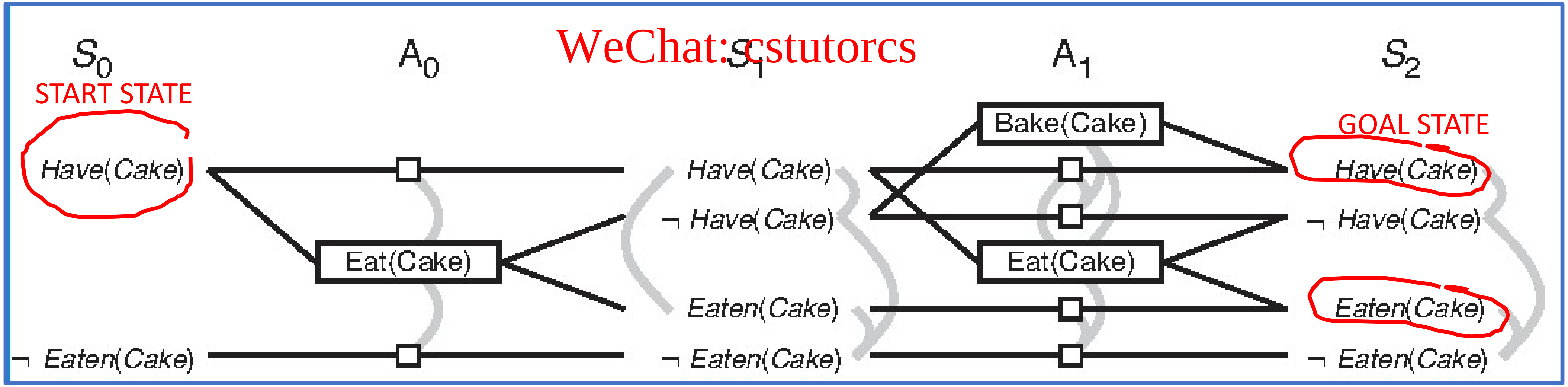
$Bake(cake)$

$Have(Cake)$



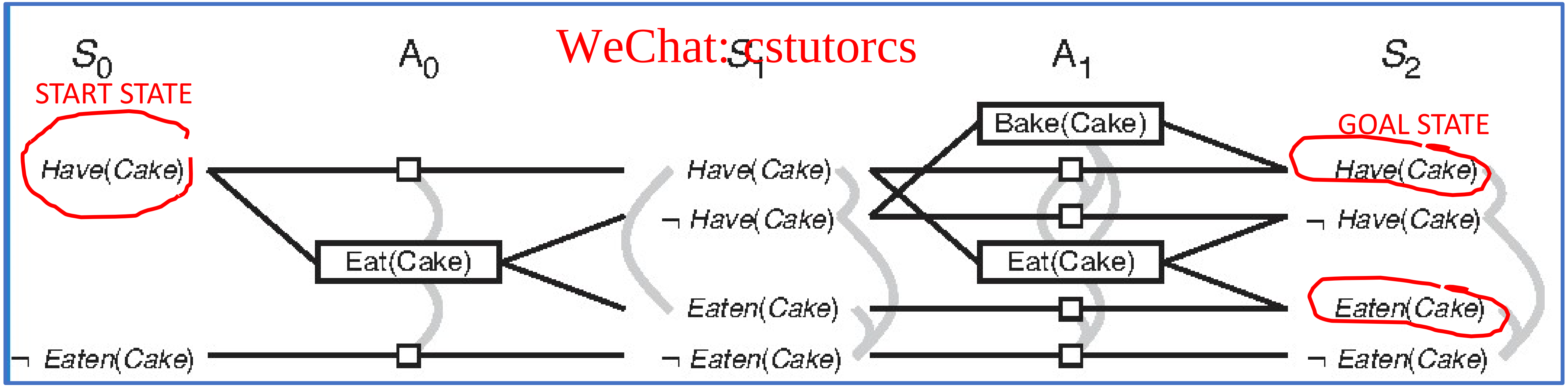
Planning graph – level S_i

- i=0: Start state + literals that “could” hold at the start
- i>0: all literals that “could” hold at S_i , depending on actions at A_{i-1}



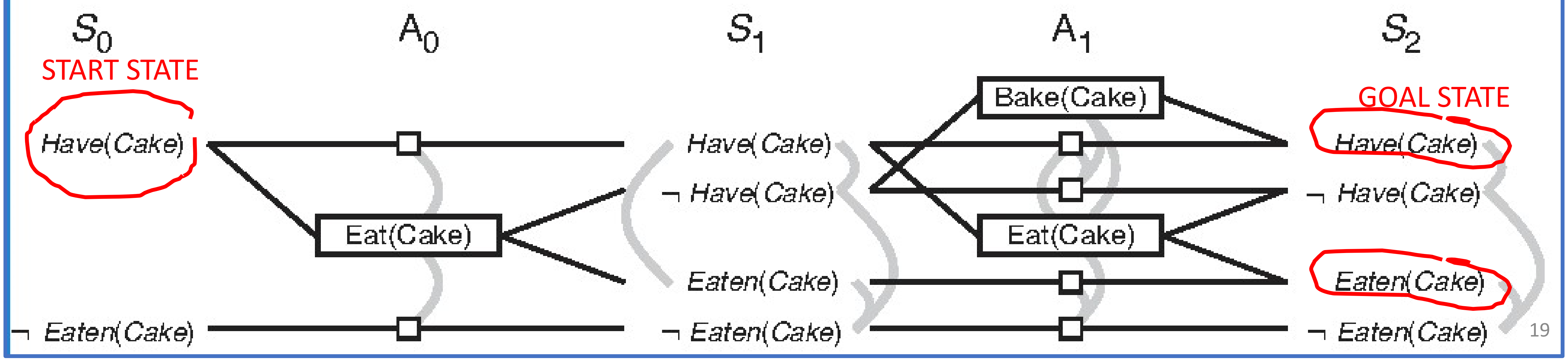
Planning graph – level A_i

- Each action at A_i connected to its precondition at S_i and effect at S_{i+1}
- No-op actions (□): a literal persists if no action negates it



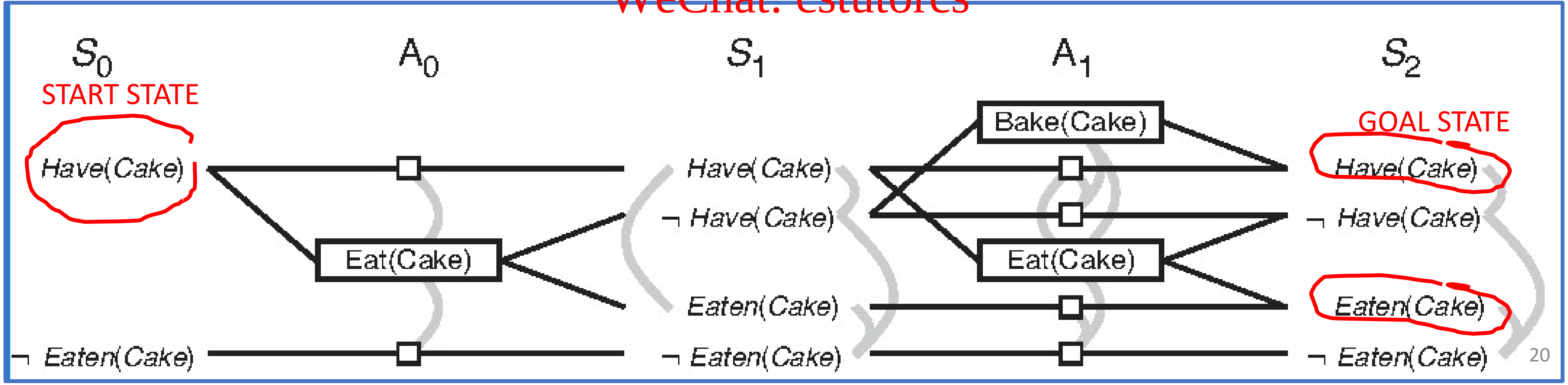
Planning graph – mutex (mutual exclusions) between actions

- Inconsistent effects
  (e.g. $Eat(Cake)$ and □)
- Interference: effect of one action= $\neg$precondition of the other
  (e.g. $Eat(Cake)$ and □)
- Competing needs: mutex preconditions
  (e.g. $Eat(Cake)$ and $Bake(Cake)$)



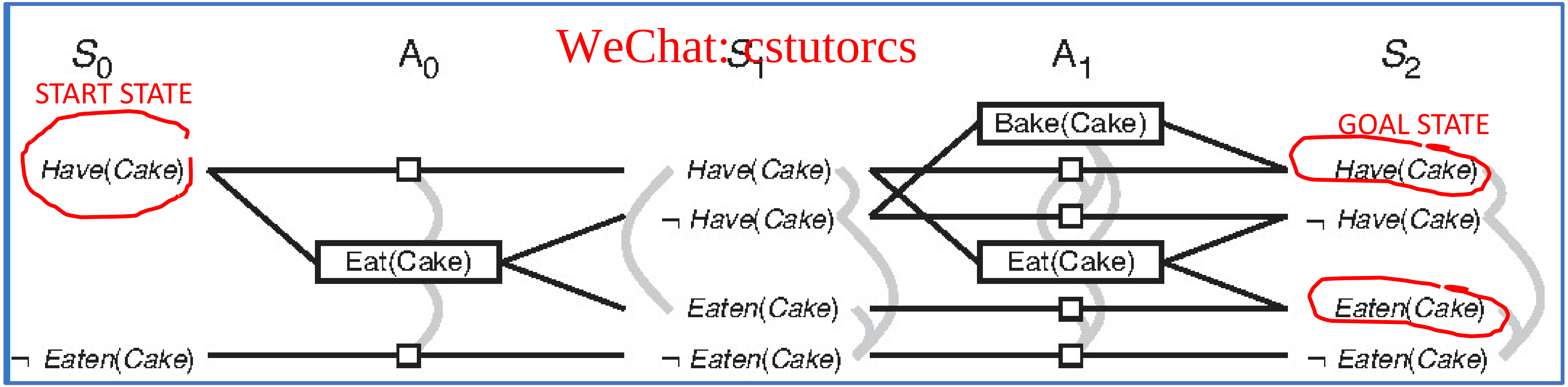
Planning graph – mutex (mutual exclusions) between literals

- Complementary literals
  (e.g. $Have(Cake)$ and $\neg Have(Cake)$)
- Inconsistent support: each possible pair of actions achieving them are mutex
  (e.g. $Have(Cake)$ and $Eaten(Cake)$ at S_1)



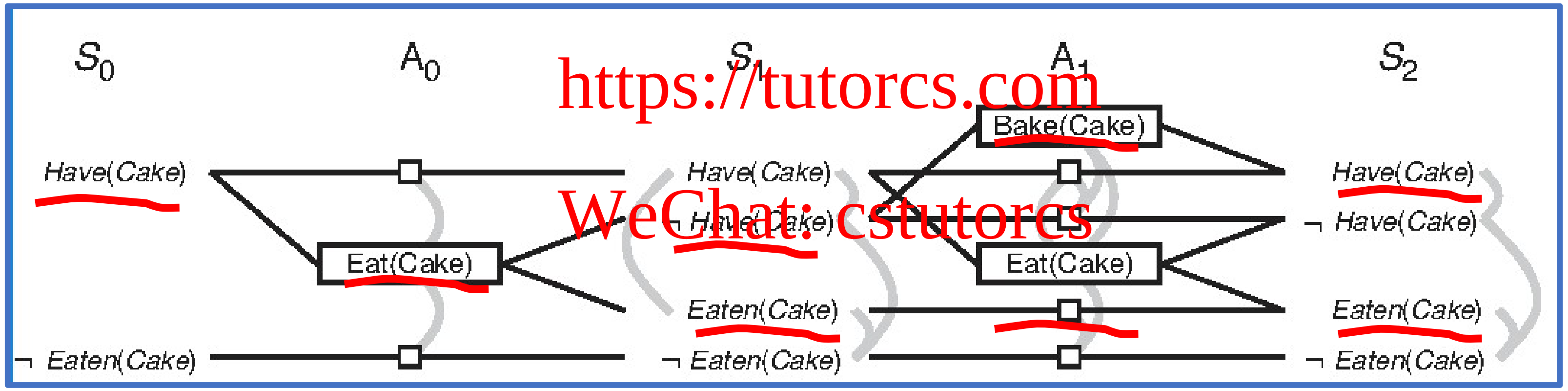
Planning graph: levelling off

$S_1 = S_2$



Graph-plan algorithm - idea

- Construct S_0 , let $i = 0$
- If goal non-mutex in S_i then **extract plan** (backwards, if any) from S_i



- Else expand planning graph (construct A_i and S_{i+1}) and increment i

Extract plan

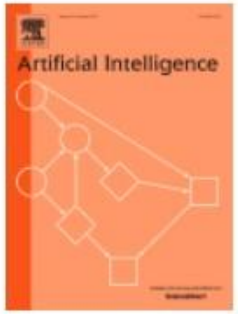
- To ***extract*** a plan P for a non-empty set G of goals at t:
  1. <u>select</u> a set A of actions at t-1 for the goals in G, given {}
  2. *extract* a plan P' for the preconditions of actions in A at t-1, and return P=AUP'
- To ***extract*** a plan P for an empty set G of goals at t:
  1. Return P={}

- To <u>select</u> a set A of actions at t-1 for the goals in a non-empty set G, given A’
  1. pick g in G and some action a at t-1 having g as add-effect and such that a is not mutex of any action in A’
  2. *select* a set A'' of actions at t-1 for the goals in G-{g}, given A'U{a}, and return A=A''U{a}
- To *<u>select</u>* a set A of actions at t-1 for the goals in an empty set G, given A'
  1. Return A={}

Omitted

- Planning and acting in the real-world (under uncertainty)
- Hierarchical planning
- Multi-agent planning

Planning – today (example – non-examinable)



ELSEVIER

Artificial Intelligence

Volume 256, March 2018, Pages 1-34

Strong temporal planning with uncontrollable durations ☆

Alessandro Cimatti [a], Minh Do [b], Andrea Micheli [a], Marco Roveri [a], David E. Smith [b]

⊞ **Show more**

https://doi.org/10.1016/j.artint.2017.11.006 Get rights and content

Planning – today (example – non-examinable)

ELSEVIER

Artificial Intelligence
Volume 252, November 2017, Pages 175-210

The MADLA planner: Multi-agent planning by combination of distributed and local heuristic search

Michal Štolba, Antonín Komenda

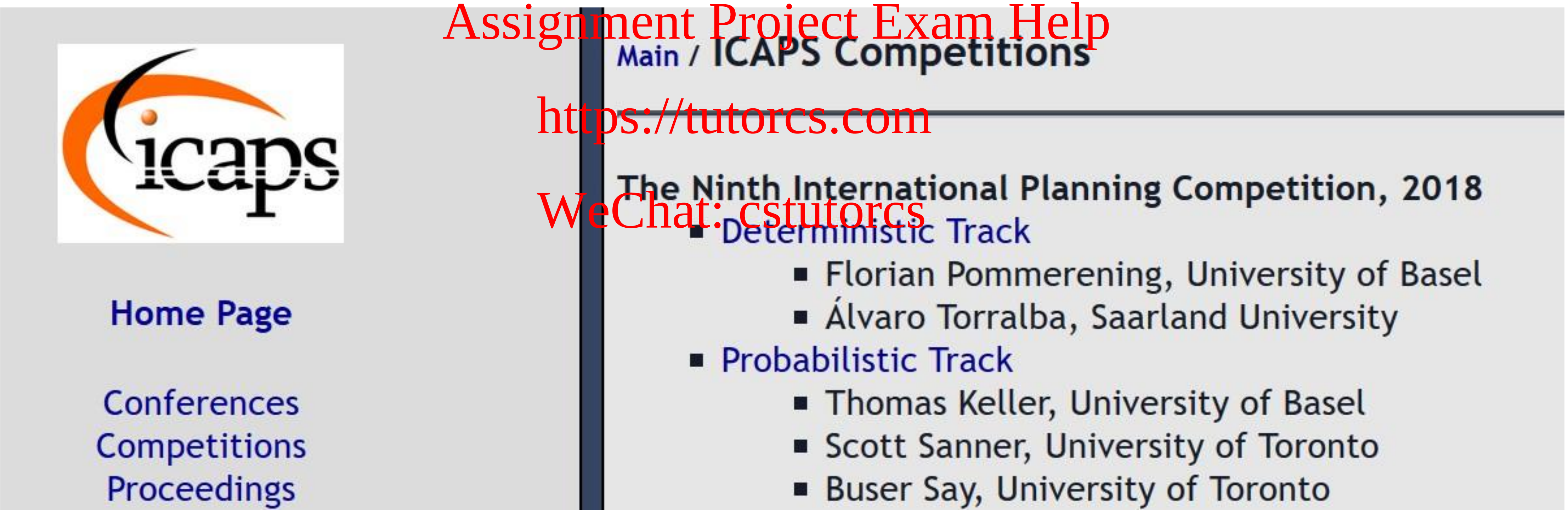
Show more

https://doi.org/10.1016/j.artint.2017.08.007

Get rights and content

Planning today (non-examinable)

International Conference on Automated Planning and Scheduling



Planning - summary

- Formulation of planning problems
- Understanding planning as search
- Tailored planning algorithms are useful
- Graph-plan as a seminal planning algorithm