Task/action planning: Classical and embedded trends

Philippe Morignot

Part I --- Classical planning

Introduction

Topics list of the European Conference on Artificial Intelligence ECAI'16:

- Autonomous Agents and Multi-agent Systems
- Constraints, Satisfiability, and Search
- Knowledge Representation, Reasoning, and Logic
- Machine Learning and Data Mining
- Natural Language Processing
- **Planning** and Scheduling
- Robotics, Perception and Vision
- Uncertainty in AI
- Web and Knowledge-based Information Systems
- Cognitive Modeling and Cognitive Architectures
- Agent-based and integrated systems
- Multidisciplinary Topics

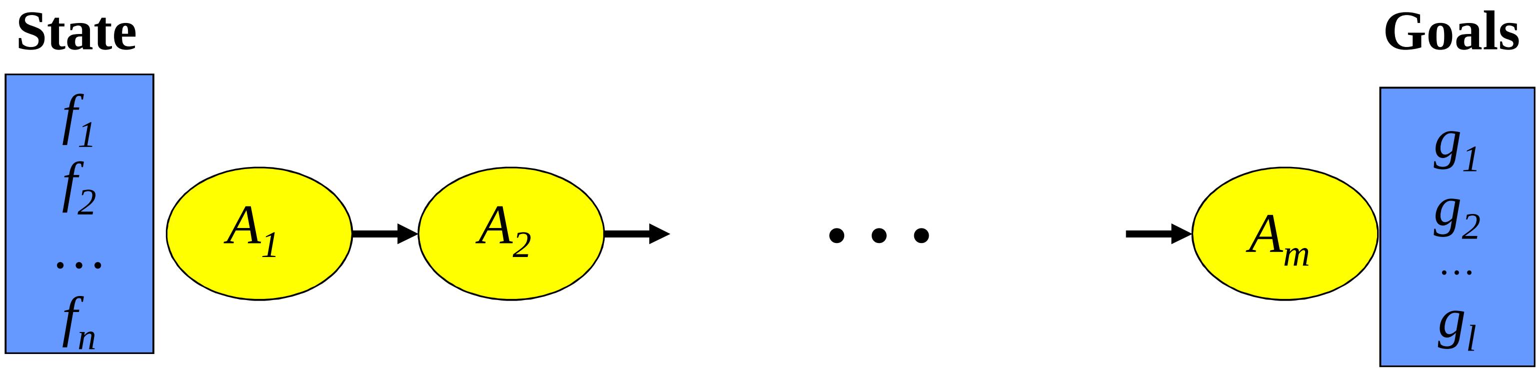
Definition (1 / 2)

- « *Given an initial state, goals and action templates, find a sequence of instantiated actions which provably lead the initial state to a state containing the goals*. »

- A.I. planning = task planning = action planning = plan synthesis = plan generation (= planning?) = *activity of finding this solution plan*.

Definition (2 / 2)

- « *Given an initial state, goals and action templates, find a sequence of instantiated actions which provably lead the initial state to a state containing the goals*. »



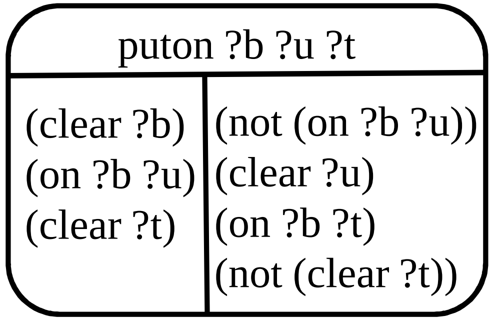
- A.I. planning = task planning = action planning = plan synthesis = plan generation (= planning?) = *activity of finding this solution plan*.

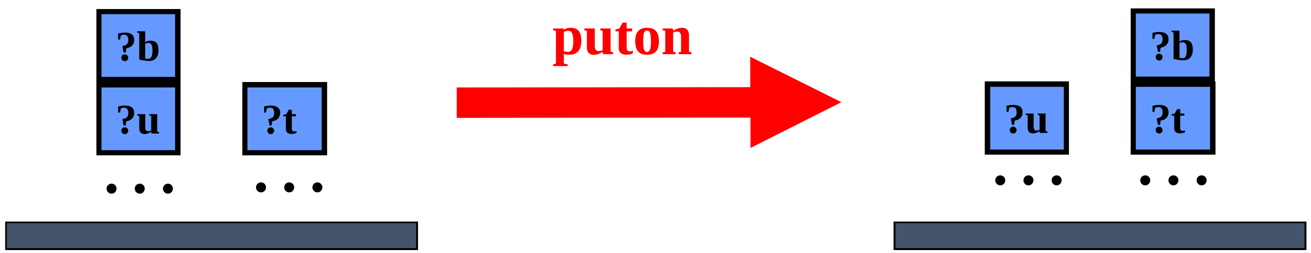
Planning Domain Definition Language (1 / 3)

- PDDL is a representation language which proposes:
  - A ***domain***: operators;
  - A ***problem***: an initial state and goals.
- An operator is composed of:
  - ***Pre-conditions***: terms which must hold for the operator to be executable;
  - ***Post-conditions*** / ***effects***: terms the truth value of which are changed by the execution of the operator, i.e., added (Post+, ADD-LIST, positive) or retracted (Post-, DELETE-LIST, negative).
- If an operator is applicable: $S_{out} = S_{in} \cup Post+ \setminus Post-$
- A term can be sometimes true and sometimes false (non monotonicity), depending on the time in the plan at which this term is considered.
  - Logical negation, e.g., (**not** (ON MOUSE PAD)).
  - *Fluent*, e.g., (ON MOUSE PAD).

Planning Domain Definition Language (2 / 3)

```
(:action puton
 :parameters (?b ?u ?t - block)
 :precondition (and (clear ?b) (on ?b ?u))
                    (clear ?t))
 :effect (and (not (on ?b ?u)) (clear ?u)
              (on ?b ?t) (not (clear ?t))))
```

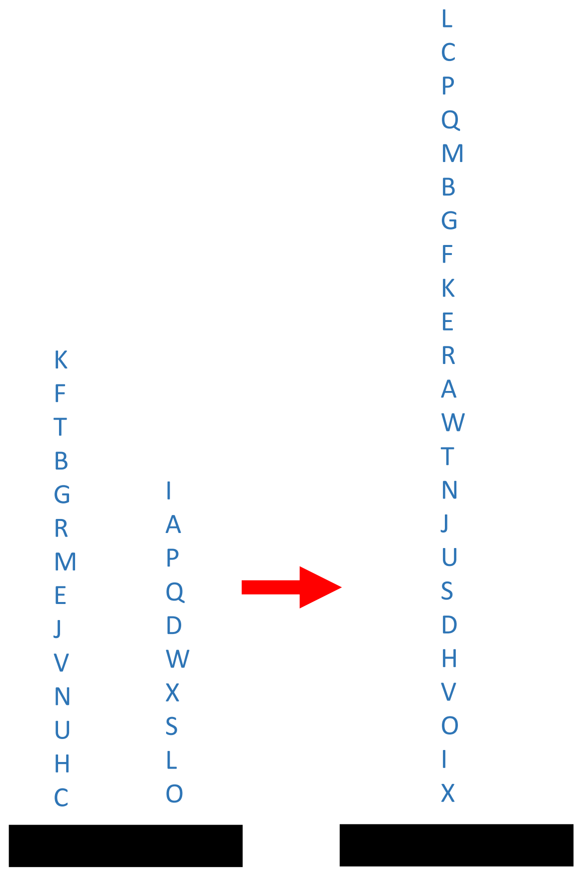




- Table? Colors? Arms? Nicks? Sizes? Conditionals? Universal quantifications?
- Qualification problem; Ramification problem.
- Frame problem: Closed world assumption.

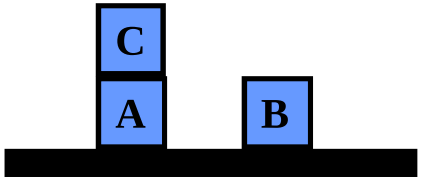
Planning Domain Definition Language (3 / 3)

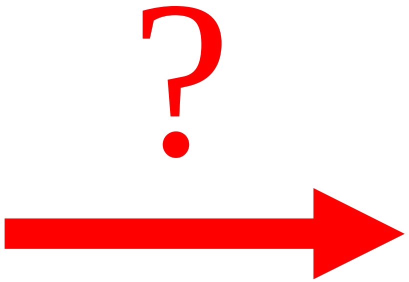
```
(define    (problem blocks-24-1)
    (:domain blocks)
    (:objects X W V U T S R Q P O N M L K J I H G F E D C A B)
    (:init
            (CLEAR K) (CLEAR I) (ONTABLE C) (ONTABLE O)
            (ON K F) (ON F T) (ON T B) (ON B G) (ON G R)
            (ON R M) (ON M E) (ON E J) (ON J V) (ON V N)
            (ON N U) (ON U H) (ON H C) (ON I A) (ON A P)
            (ON P Q) (ON Q D) (ON D W) (ON W X) (ON X S)
            (ON S L) (ON L O) (HANDEMPTY))
    (:goal (and
            (ON L C) (ON C P) (ON P Q) (ON Q M) (ON M B)
            (ON B G) (ON G F) (ON F K) (ON K E) (ON E R)
            (ON R A) (ON A W) (ON W T) (ON T N) (ON N J)
            (ON J U) (ON U S) (ON S D) (ON D H) (ON H V)
            (ON V O) (ON O I) (ON I X))))
```



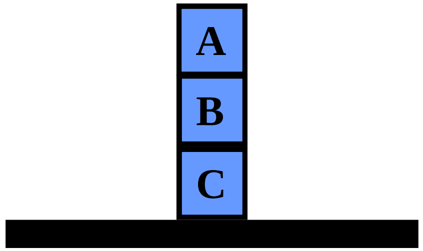
The anomaly of Gerald Jay Sussman (1 / 2)

(clear C)
(on C A)
(on A table)
(clear B)
(on B table)
(clear table)





(on A B)
(on B C)

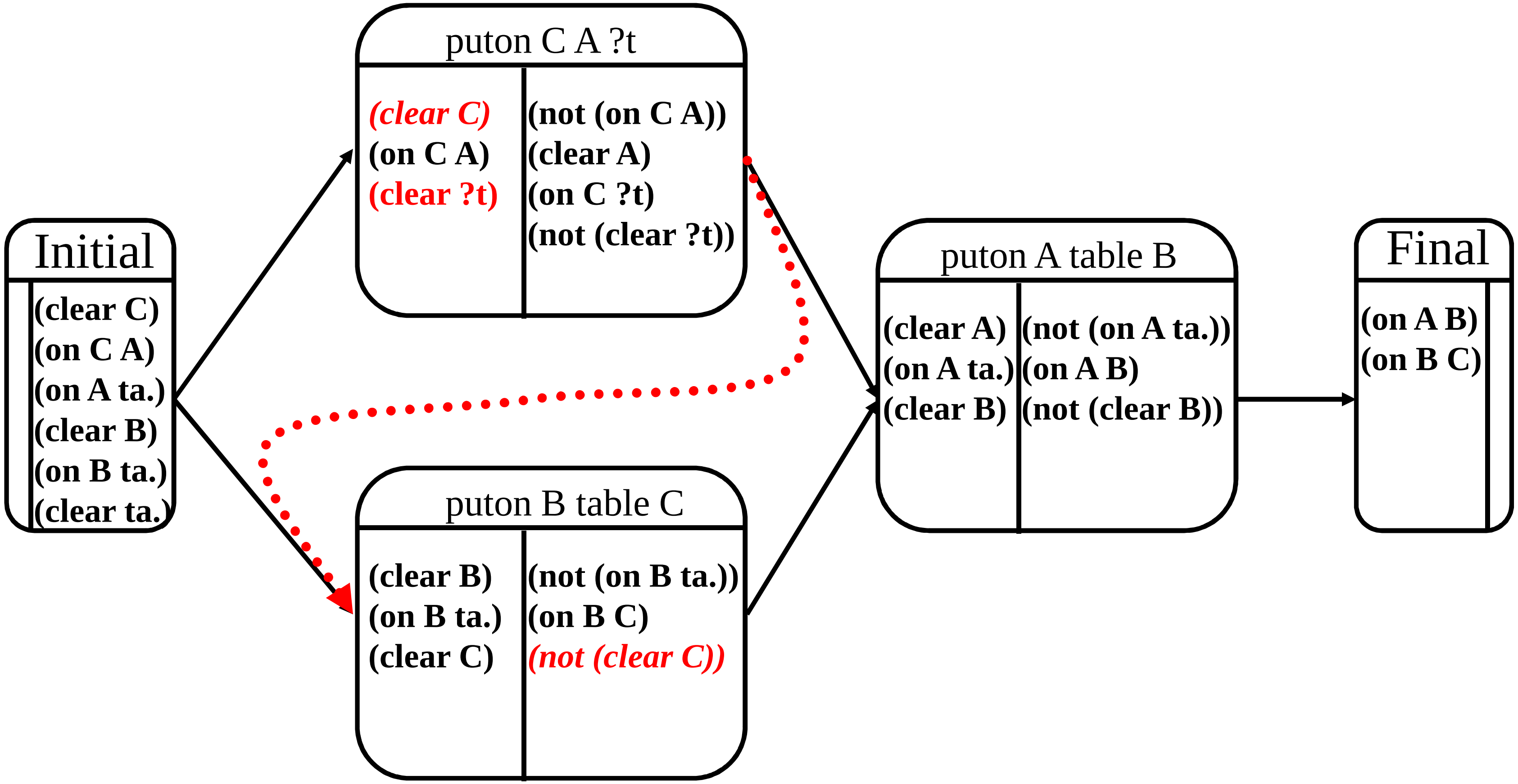


with operator :

puton ?b ?u ?t	
(clear ?b) (on ?b ?u) (clear ?t)	(not (on ?b ?u)) (=> (<> ?u table) (clear ?u)) (on ?b ?t) (=> (<> ?t table) (not (clear ?t)))

P. Morignot – 06/01/2012

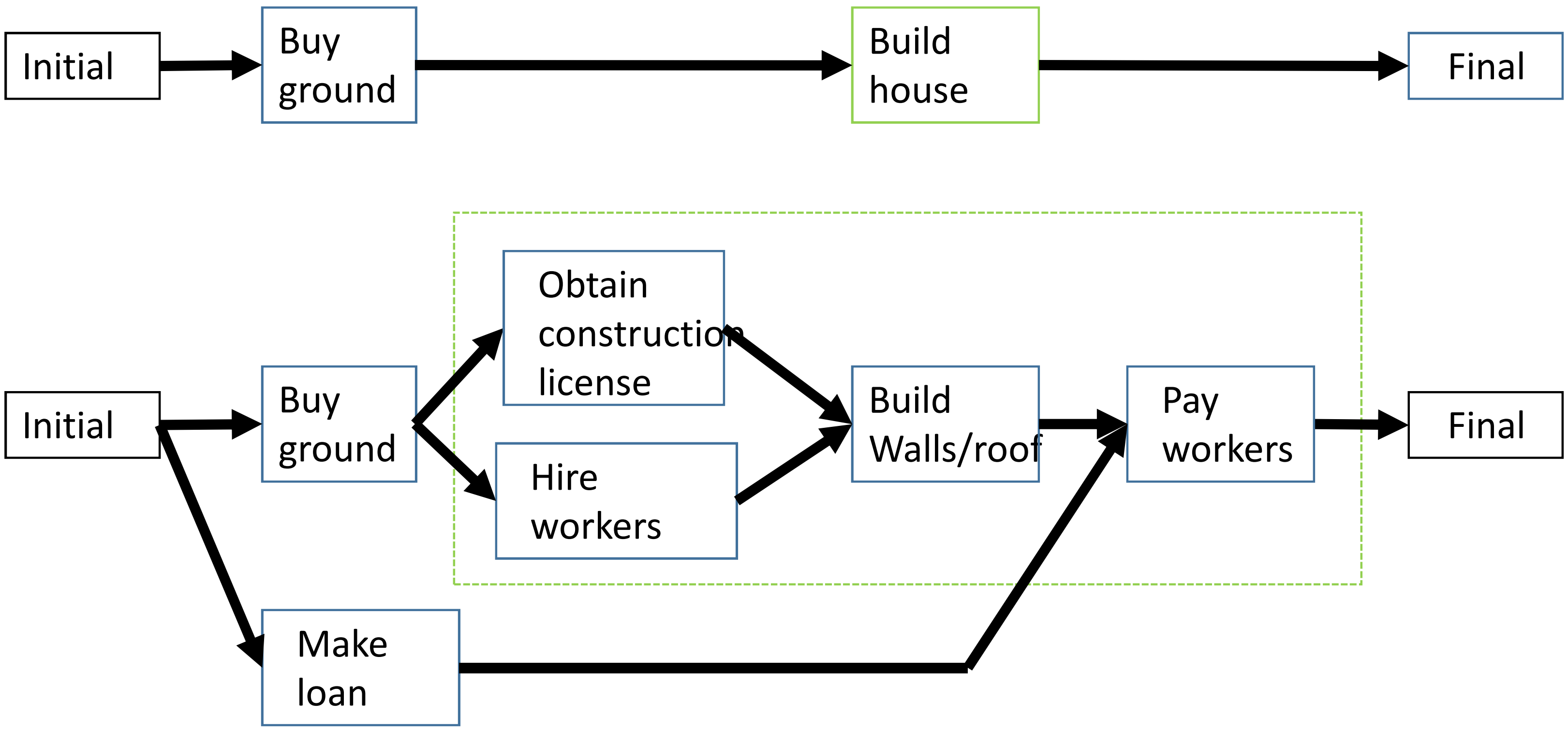
The anomaly of Gerald Jay Sussman (2 / 2)



Planners

- Using forward search in a state space (J. Hoffman, H. Geffner).
- Using backward search in a state space (M. Helmert).
- Using (forward) search in a plan space (A. Barrett).
- Using evolutionnary algorithms (M. Schoenauer).
- Using temporal logic (P. Doherty).
- Using mixed integer programming (D. Nau).
- Using constraint programming (P. Laborie, V. Vidal).
- Using propositional satisfiability (H. Kautz & B. Selman, J. Rintanen).

Hierarchical Task Network



Planner using SAT solvers: Principle

1. Set the length of the plan to *n (= 1* initially*)*
2. Encode the planning problem of size *n* as a propositional formula:

   $$initial\_state \wedge all\_plans\_n \wedge goals$$

3. Run a SAT solver
4. IF solution found THEN decode   *// SUCCESS*
5. Increment *n*

- Improvement: Try plan lengths in parallel.

Planner using SAT solvers: Encoding

- Goals: $on(A,B)@T \wedge on(B,C)@T$
- Initial state: $clear(C)@0 \wedge on(C,A)@0 \wedge clear(B)@0$
  $(\wedge \neg on(A,C)@0 \wedge \neg on(A,B)@0 \wedge \neg on(B,C)@0 \wedge \neg on(B,A)@0$
  $\wedge \neg on(C, B)@0 \wedge \neg clear(A)@0)$ *// closed world assumption*
- Axiom schemas on preconditions:
  $\forall \boldsymbol{x}, \forall \boldsymbol{y}, \forall \boldsymbol{z}, \forall \boldsymbol{t}$:
  $puton(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})@\boldsymbol{t} \Rightarrow on(\boldsymbol{x},\boldsymbol{y})@\boldsymbol{t} \wedge clear(\boldsymbol{x})@\boldsymbol{t} \wedge clear(\boldsymbol{z})@\boldsymbol{t}$
- Axiom schemas on effects:
  $\forall \boldsymbol{x}, \forall \boldsymbol{y}, \forall \boldsymbol{z}, \forall \boldsymbol{t}$:
  $on(\boldsymbol{x},\boldsymbol{y})@\boldsymbol{t} \wedge clear(\boldsymbol{x})@\boldsymbol{t} \wedge clear(\boldsymbol{z})@\boldsymbol{t} \wedge puton(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})@\boldsymbol{t} \Rightarrow clear(\boldsymbol{y})@\boldsymbol{t}+1 \wedge on(\boldsymbol{x},\boldsymbol{z})@\boldsymbol{t}+1$
- Axiom schemas for one operator at a time:
  $\forall \boldsymbol{x}, \forall \boldsymbol{y}, \forall \boldsymbol{y'}, \forall \boldsymbol{z}, \forall \boldsymbol{z'}, \forall \boldsymbol{t} \,/\, \boldsymbol{y} <> \boldsymbol{y'} \wedge \boldsymbol{z} <> \boldsymbol{z'}$:
  $\neg (puton(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})@\boldsymbol{t} \wedge puton(\boldsymbol{x} ,\boldsymbol{y'} ,\boldsymbol{z'})@\boldsymbol{t})$
- Frame axiom schemas:
  $\forall p, \forall t$: $p@(t+1) \Rightarrow (p@t \vee a_1^{p}@t \vee \ldots \vee a_n^{p}@t)$
  $\neg p@(t+1) \Rightarrow (\neg p@t \vee a_1^{\neg p}@t \vee \ldots \vee a_n^{\neg p}@t)$

Applications

- Generate scenarios for a mobile robot (STRIPS, Richard Fikes, 1971).
- Advise the disassembly of a car engine (NOAH, Earl Sacerdoti, 1974).
- Organize the logistics for the military invasion of Iraq (SIPE, David Wilkins, 1980).
- Reactivate the electronic components of a spatial probe flying around Jupiter (DEEP SPACE, approx. 2000).
- Debug a xerox machine.
- Animate characters in a video game (Eric Jacopin, 2010).
- Interactive story telling (Marc Cavazza, 2010).
- …

History

- 1971: STRIPS by Richard Fikes.
- 1977: NOAH by Earl Sacerdoti
- 1981: MOLGEN by Mark Stefik
- 1986: IxTeT by Malik Ghallab.
- 1986: SIPE by David Wilkins.
- 1987: TWEAK by David Chapman.
- 1991: SNLP by Mac Allister & Rosenblitt.
- 1992: UCPOP by Anthony Barrett & Daniel Weld.
- 1992: BLACKBOX/SATPLAN by Henry Kautz & Bart Selman.
- 1997: GRAPHPLAN by Avrim Blum & Merrick Furst.
- 2000: HSP by Hector Geffner.
- 2000: YAHSP by Vincent Vidal.
- 2001: FF by Jörg Hoffmann,
- 2005: CPT by Vincent Vidal.
- 2007: DAE by Marc Schoenauer.

References

- **Stuart Russell, Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2010, 3rd edition. Chapiter 11.**
- **Malik Ghallab, Dana Nau, Paolo Traverso. *Automated Planning: Theory and Practice*. Morgan Kaufmann, San Mateo, CA, May 04, 635 pages.**
- **PDDL 3.1.: see Wikipedia.**
- **Conferences:**
  - **International Conference on Automated Planning and Scheduling (ICAPS). http://www.icaps.org**
  - **International Joint Conference on A.I. (IJCAI). http://www.ijcai.org**
  - **European Conference on A.I. (ECAI). http://www.ecai.org**
  - **National Conference on A.I. (AAAI). http://www.aaai.org**
- **Journals:**
  - **A. I. Journal (AIJ). http://www.elsevier.com/wps/find/journaldescription.cws_home/505601/description#description**
  - **Journal of A.I. Research (JAIR). http://www.jair.org/**

Demonstration --- Constraint Programming Temporal planner (CPT).

Part II --- Embedded planning

Introduction

Assumptions of classical planning

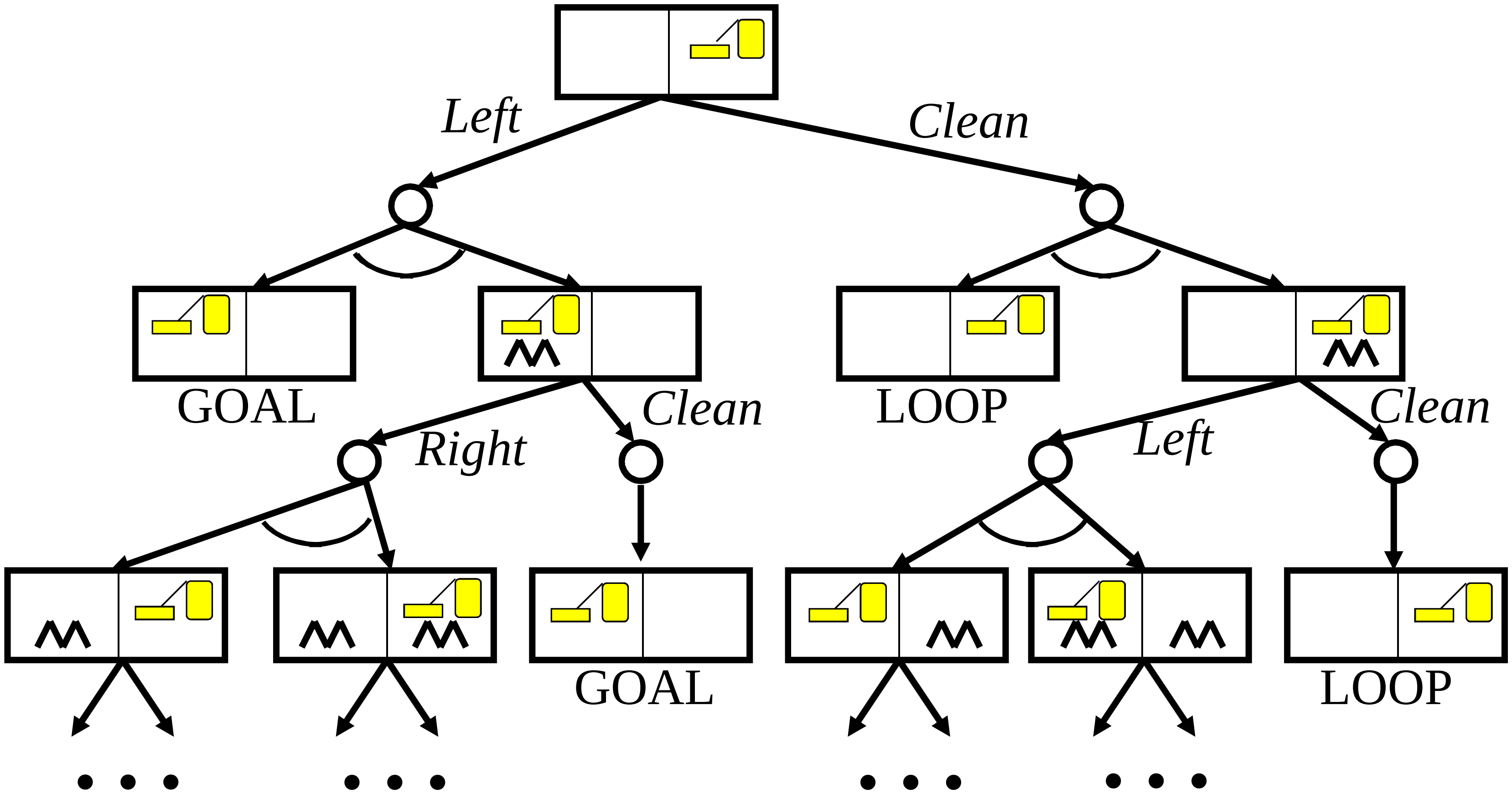
- Hypothesis 1: The agent is the unique cause of change.
- Hypothesis 2: The environment is totally observable, the agent has perfect knowledge of the environment.
- …

- What if the environment is dynamic, e.g., includes other agents?
- What if the agent partially knows its environment (perception) ?

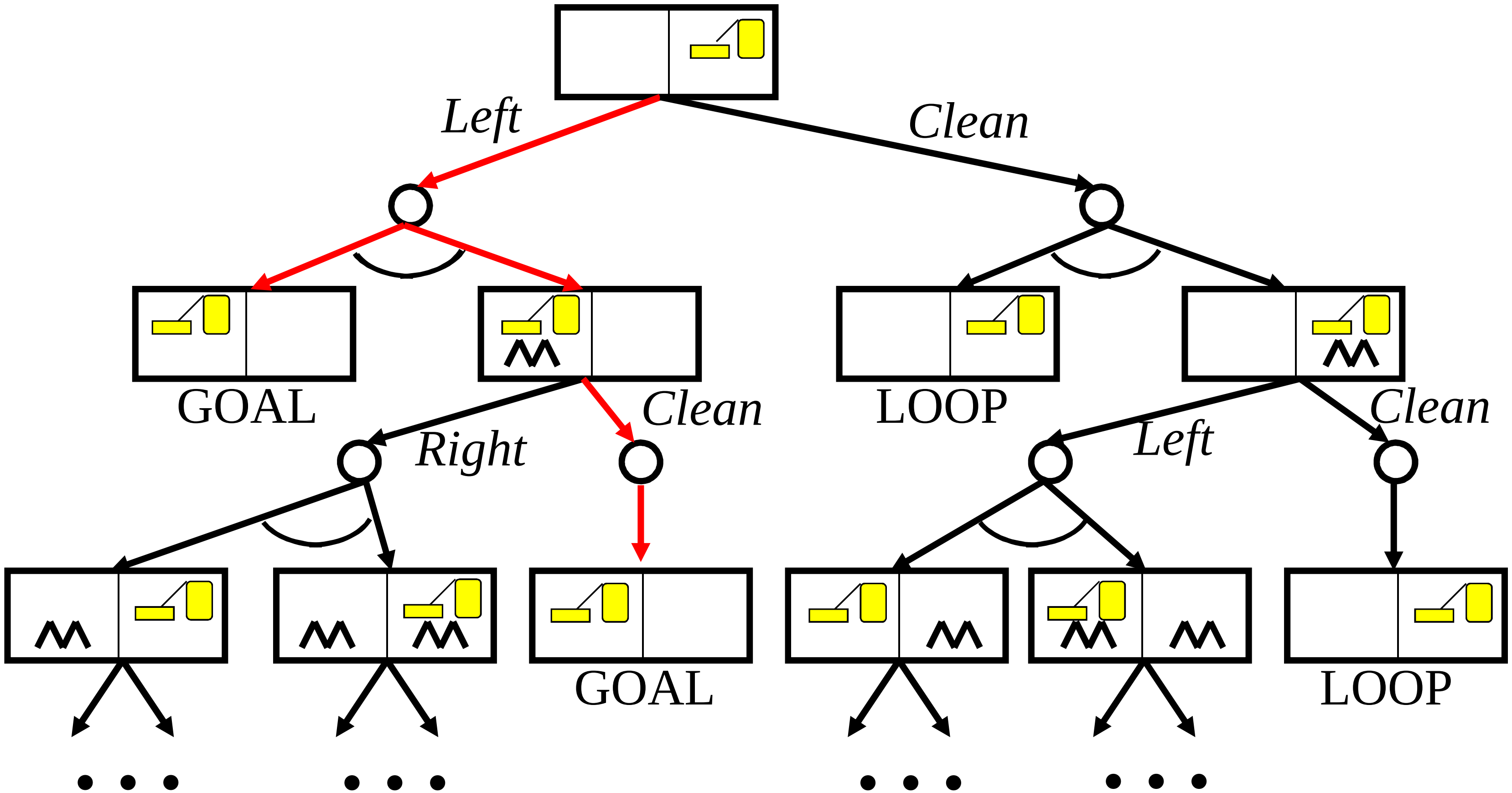
Conditionnal Planning (1 / 3)

- An operator can fail (non-determinism).
- At some times in the plan, observe what happens in the environment.
- Conditionnal and disjunctive effects.
- A conditionnal plan [A_1 ; A_2 ; … ; A_n] is a tree composed of steps
  IF *<test>* THEN *<plan-T>* ELSE *<plan-F>*
- <u>Example:</u> Domain of the vacuum cleaner in a two-room apartment.
  - Operators: Left, Right, Clean.
  - Fluents: AtLeft, AtRight, LeftClean, RightClean.
  - Rules: Cleaning a clean room can drop dust; Moving to a clean room can drop dust.
  - *Action(Left, Pre: AtRight, Post: AtLeft* $\vee$ *(AtLeft* $\wedge$ *IF LeftClean THEN* $\neg$ *LeftClean))*

Conditionnal Planning (2 / 3)



Conditionnal Planning (3 / 3)



Online Re-planning (1 / 19)

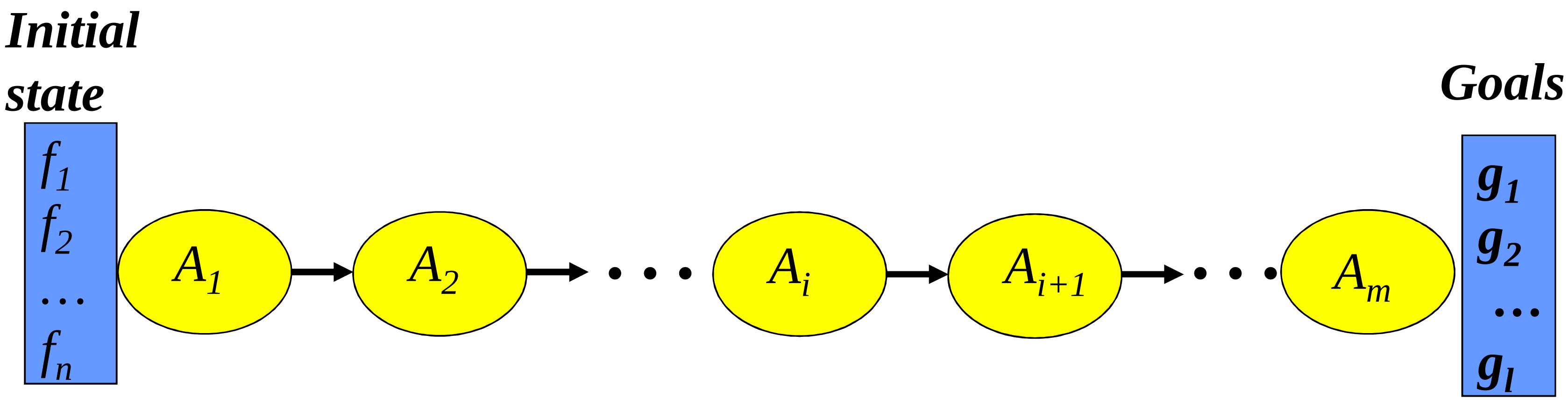
Initial state

f_1
f_2
…
f_n

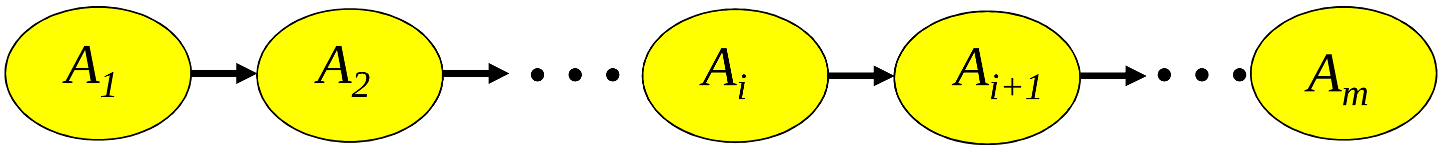
Goals

$\boldsymbol{g_1}$
$\boldsymbol{g_2}$
…
$\boldsymbol{g_l}$

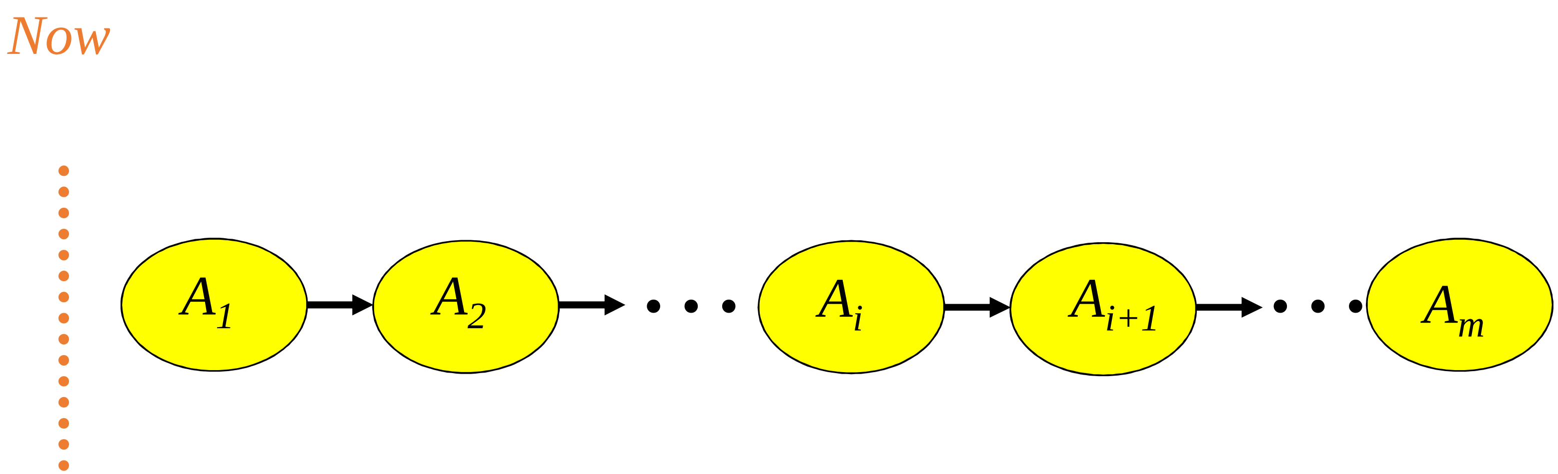
Online Re-planning (2 / 19)



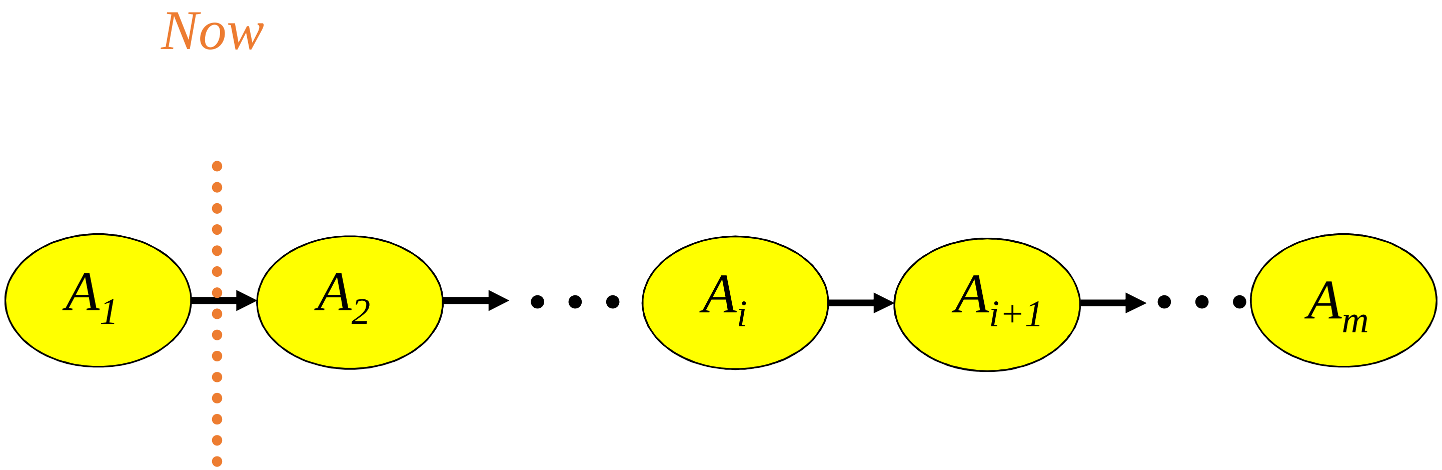
Online Re-planning (3 / 19)



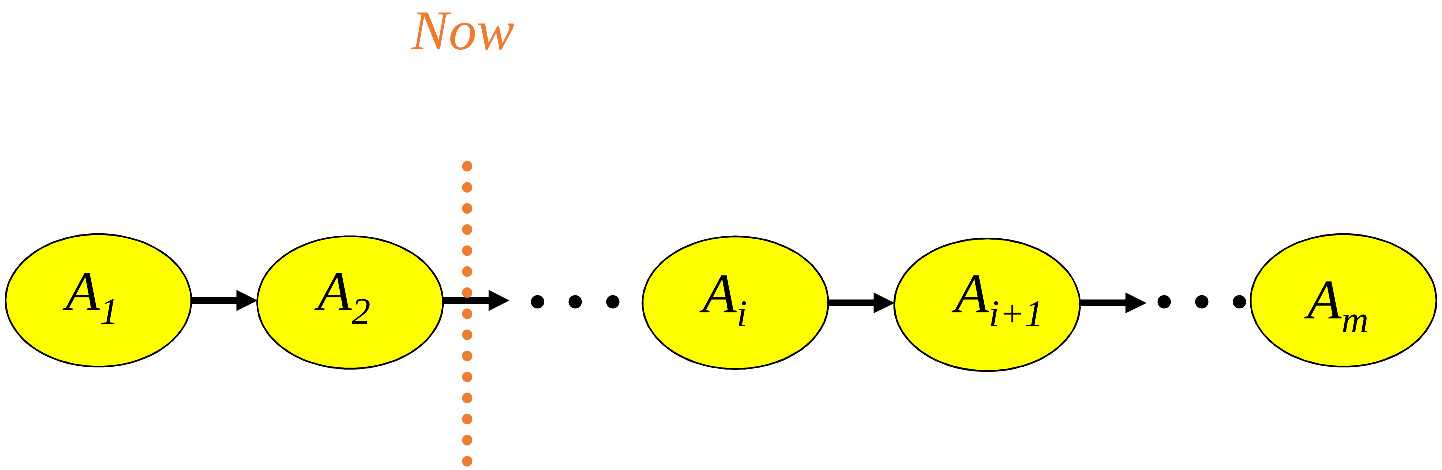
Online Re-planning (4 / 19)



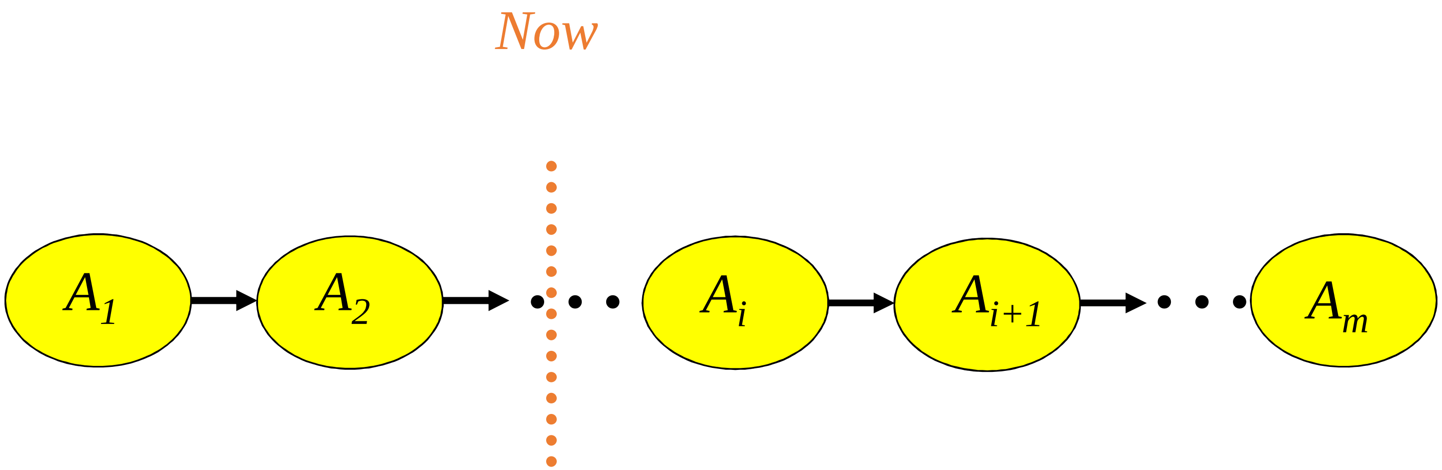
Online Re-planning (5 / 19)



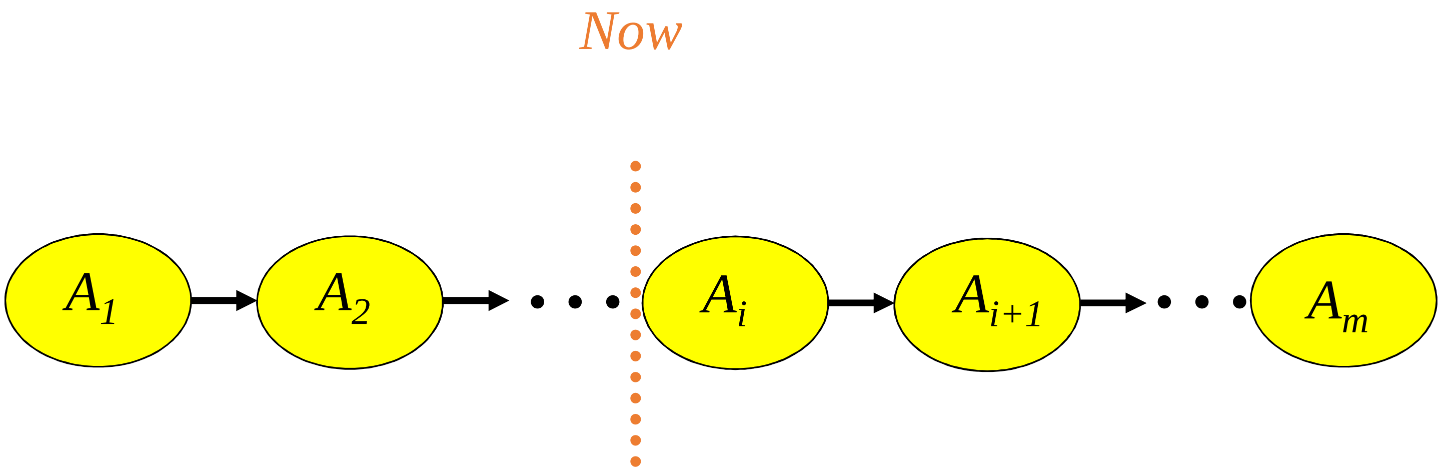
Online Re-planning (6 / 19)



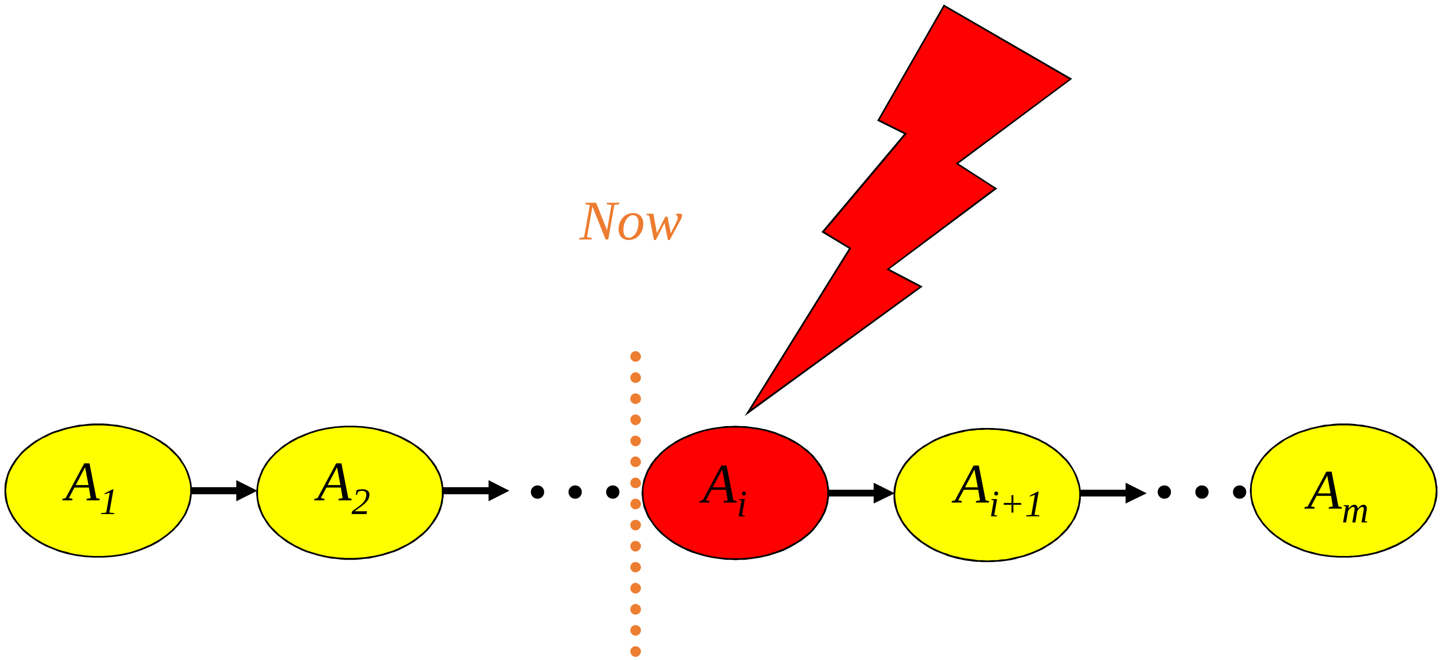
Online Re-planning (7 / 19)



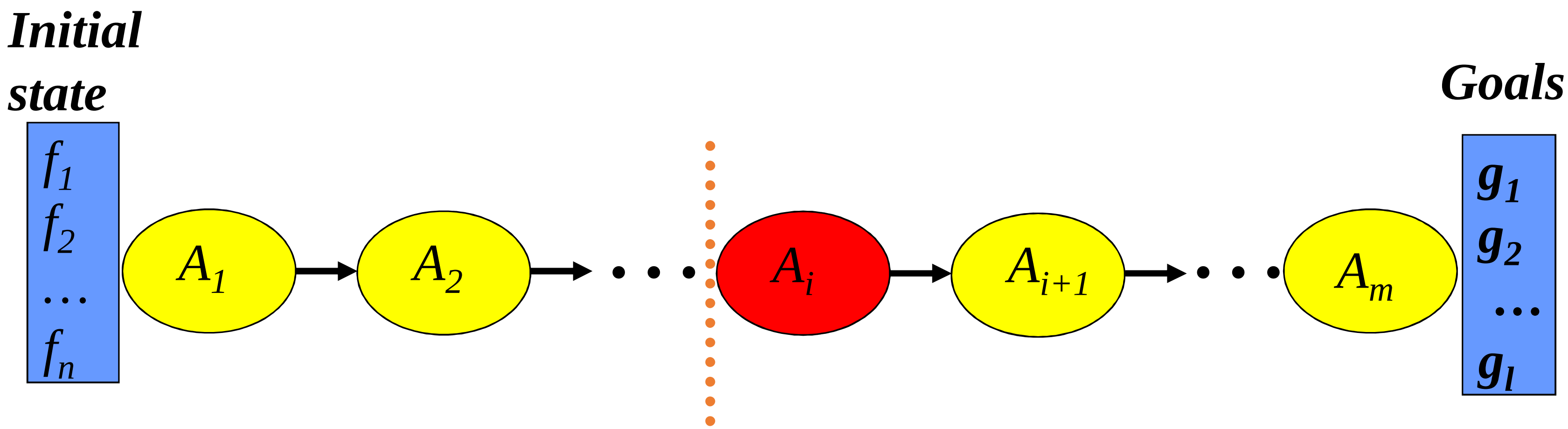
Online Re-planning (8 / 19)



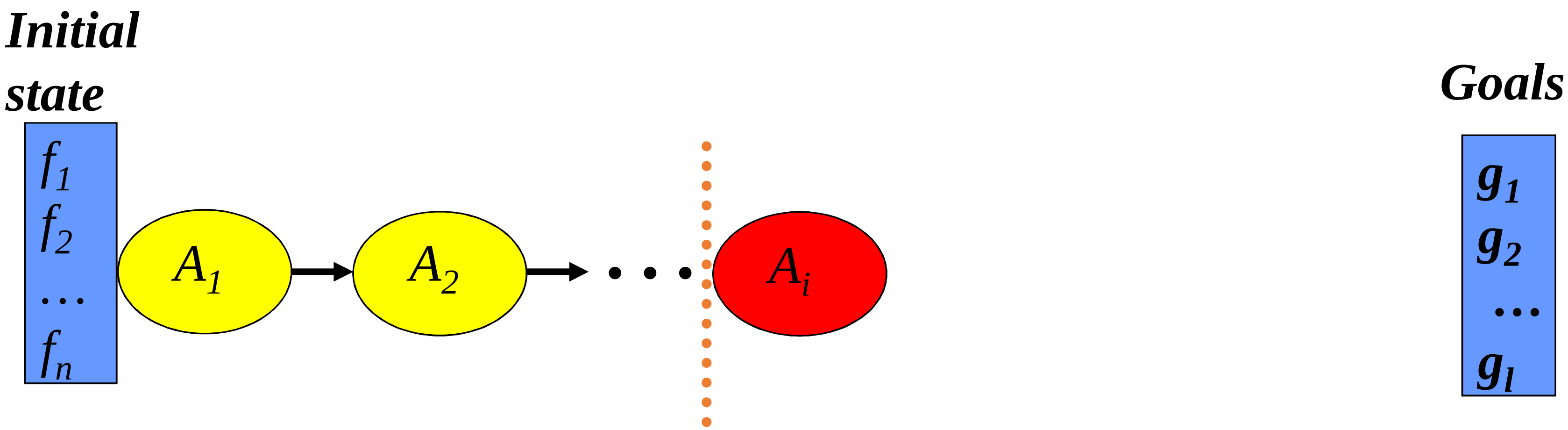
Online Re-planning (9 / 19)



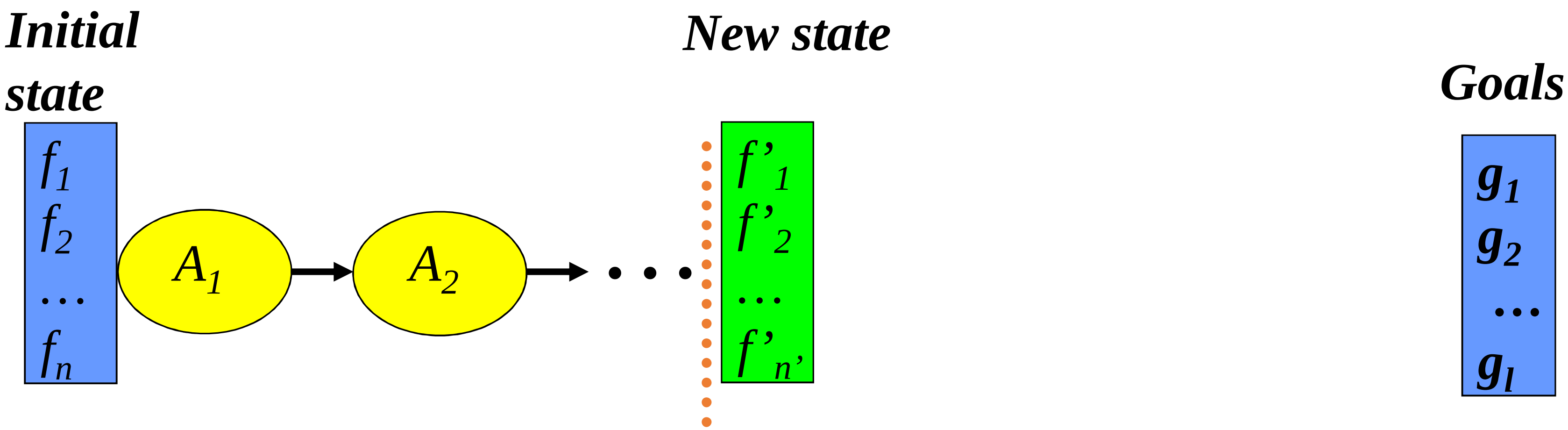
Online Re-planning (10 / 19)



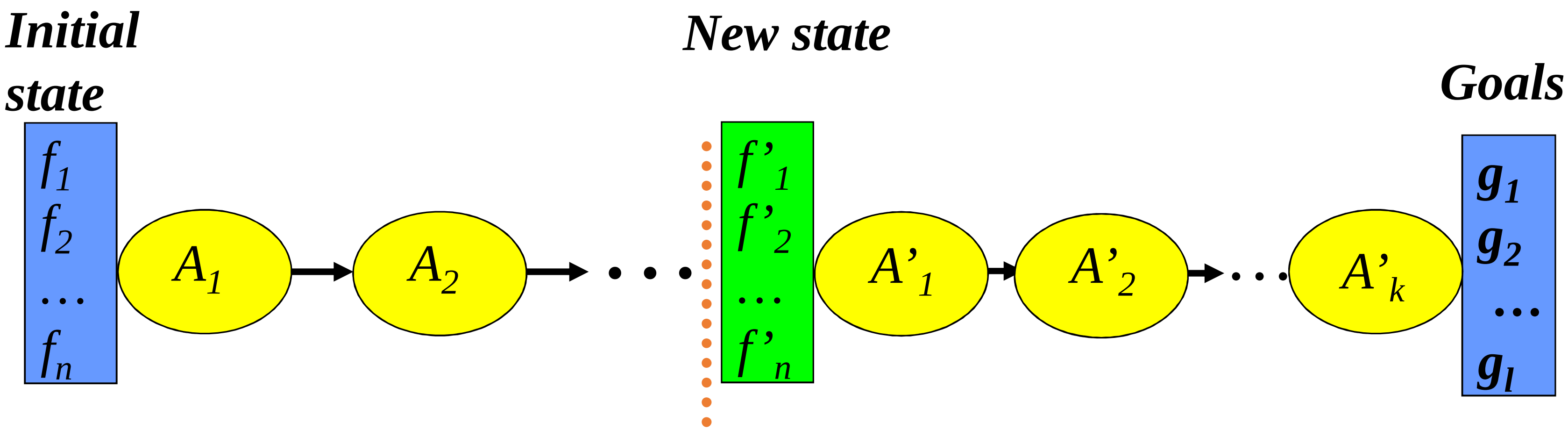
Online Re-planning (11 / 19)



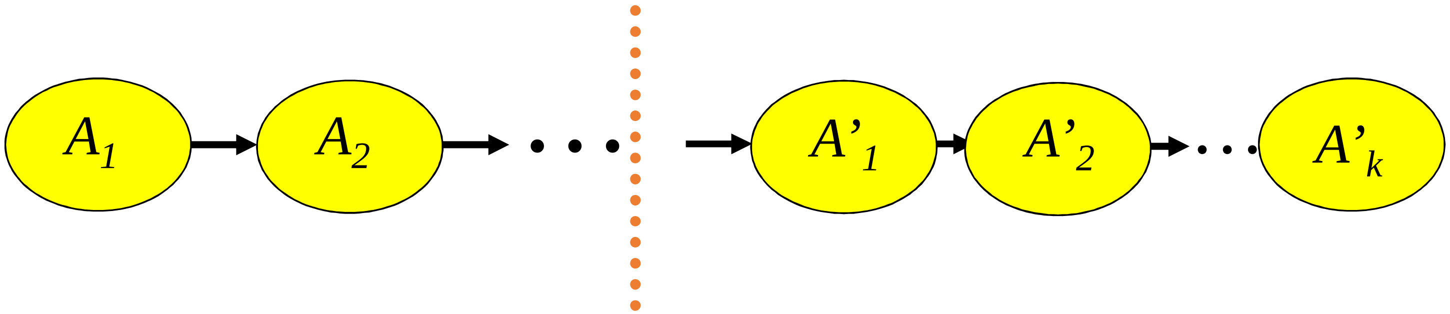
Online Re-planning (12 / 19)



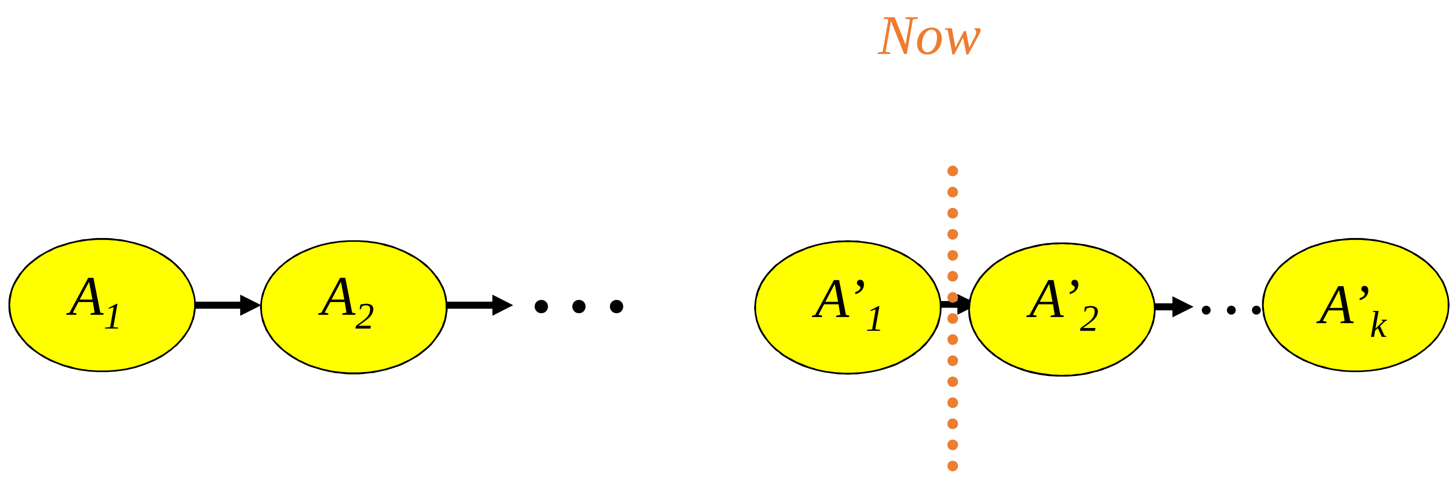
Online Re-planning (13 / 19)



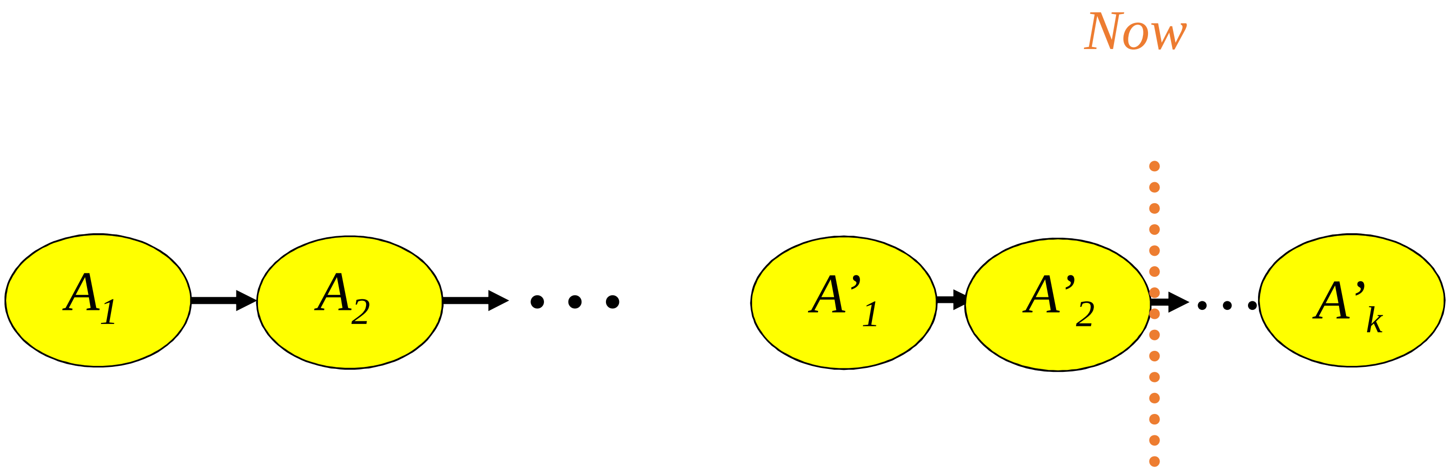
Online Re-planning (14 / 19)



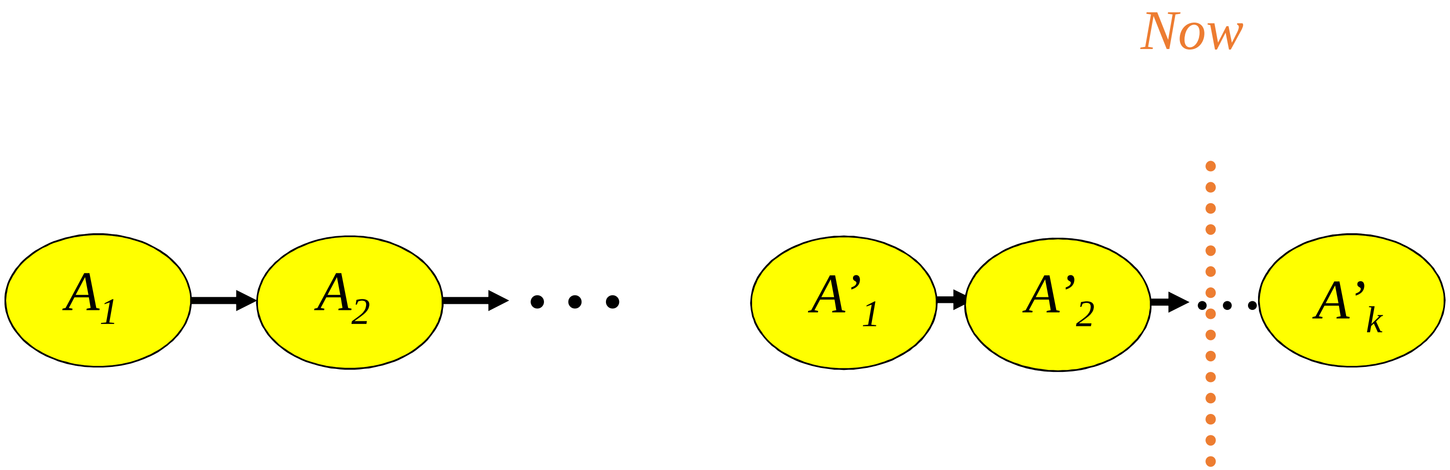
Online Re-planning (15 / 19)



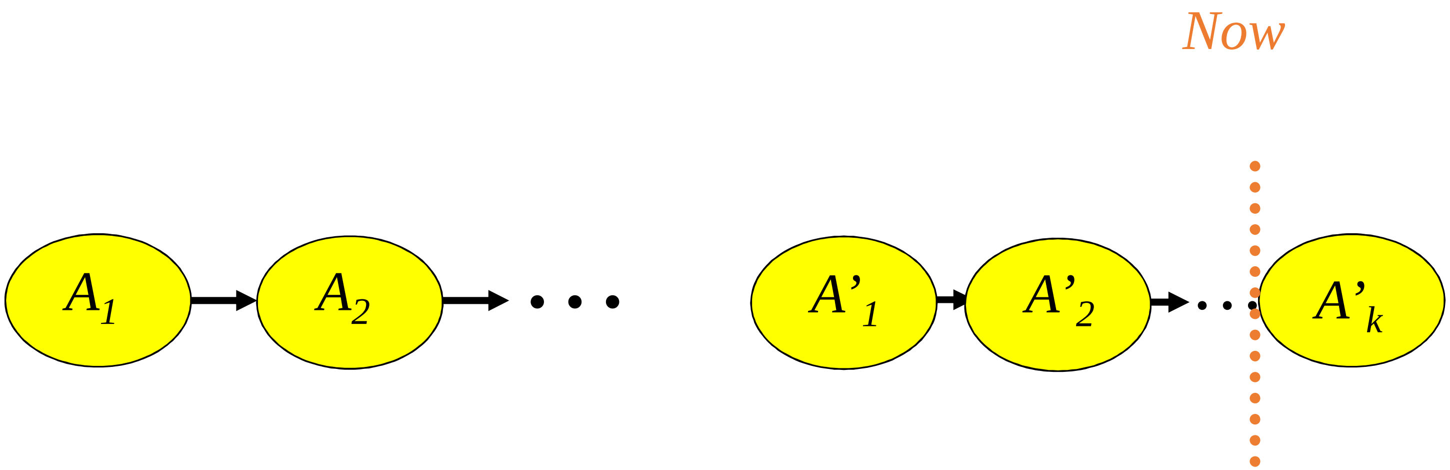
Online Re-planning (16 / 19)



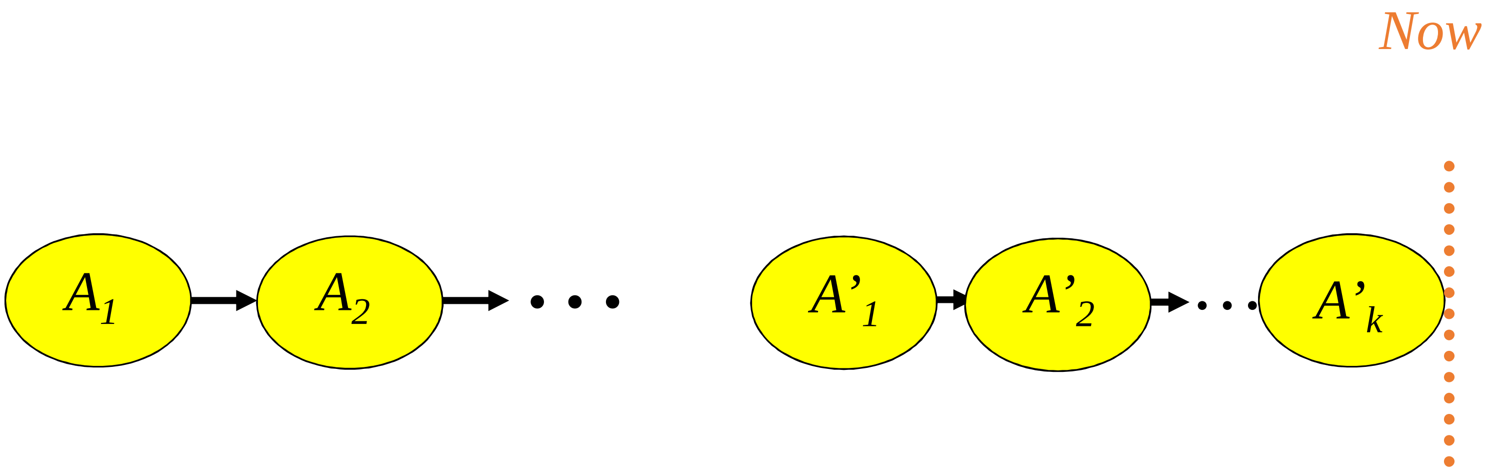
Online Re-planning (17 / 19)



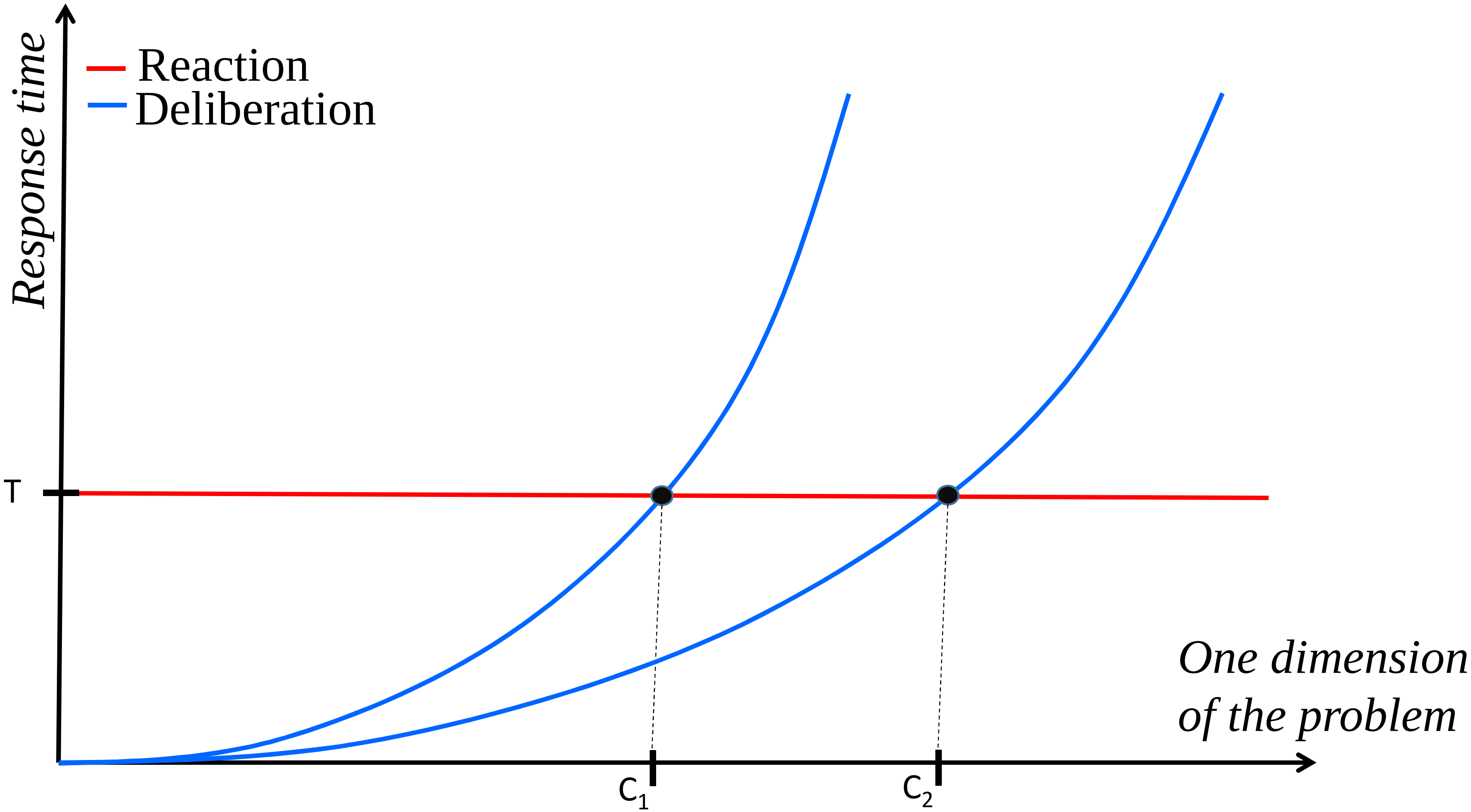
Online Re-planning (18 / 19)



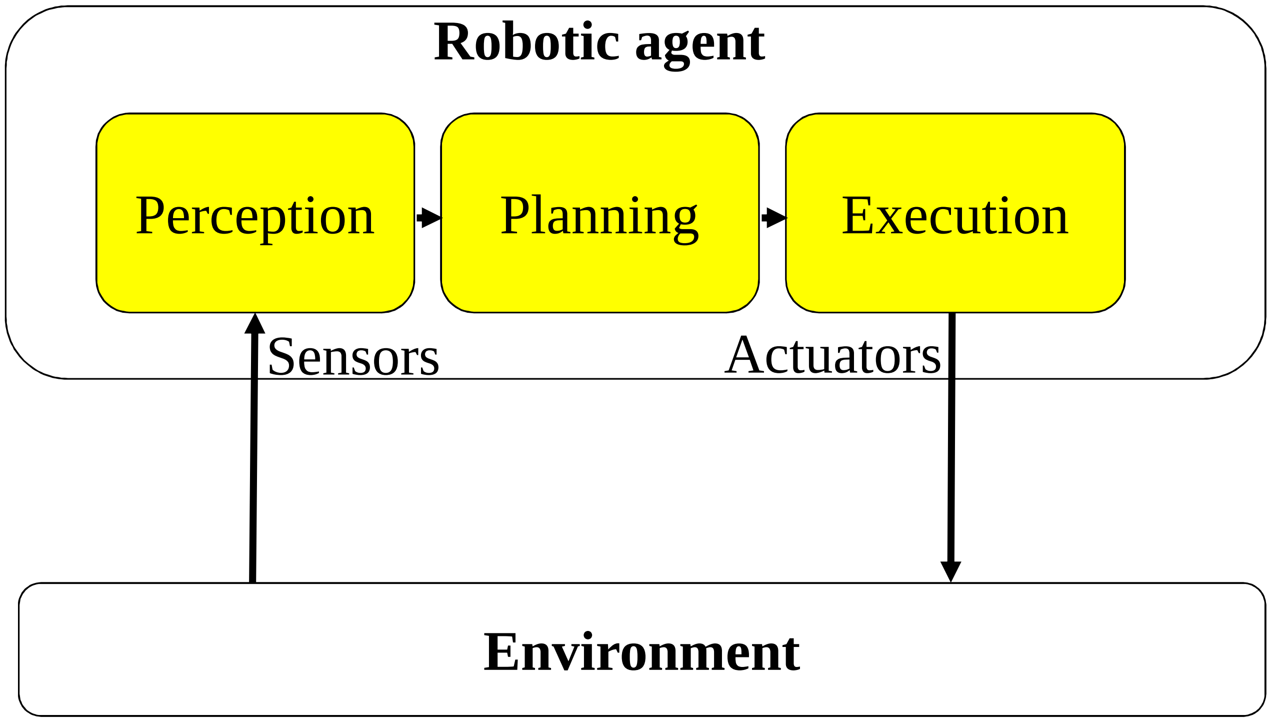
Online Re-planning (19 / 19)



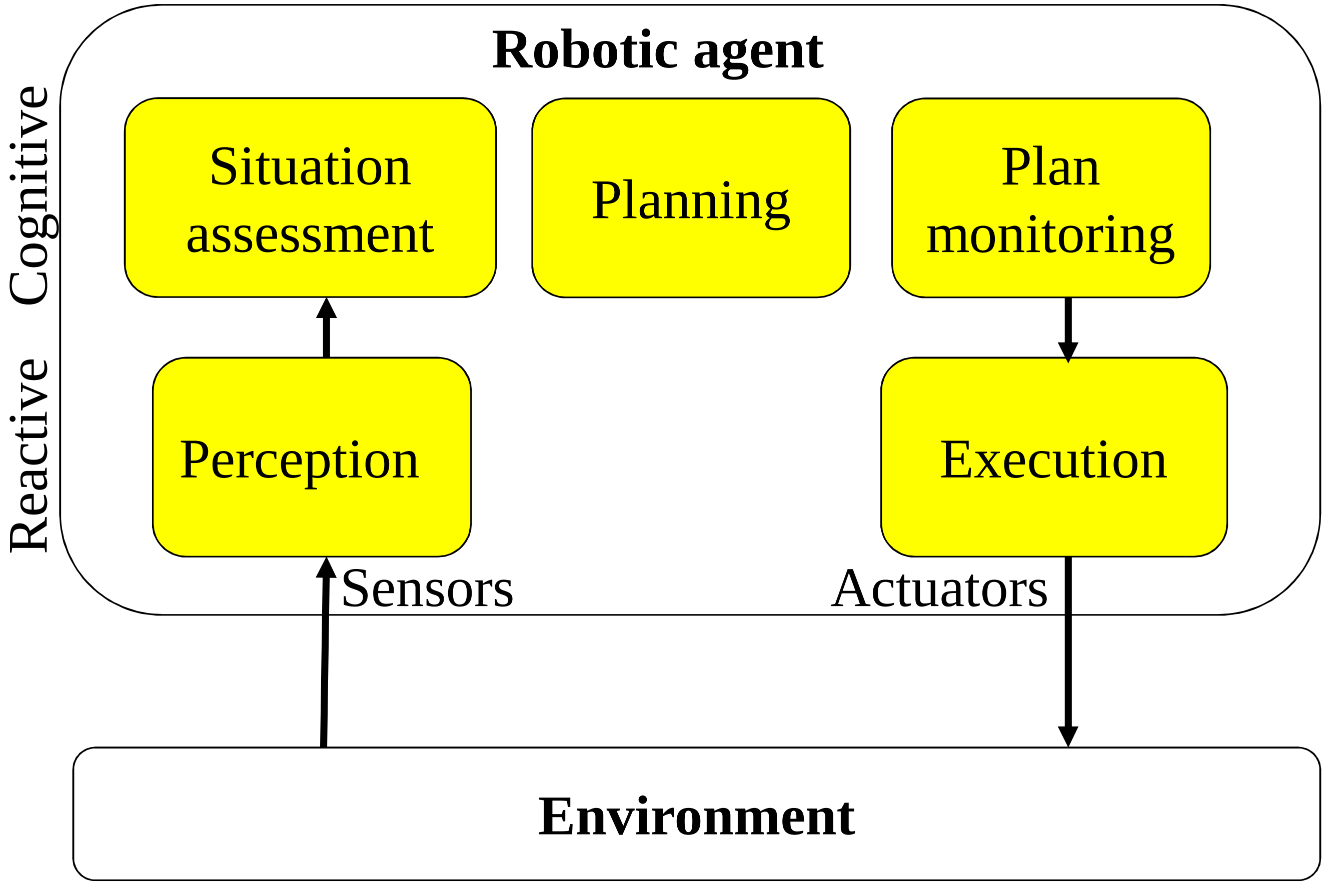
Limitation of classical planning



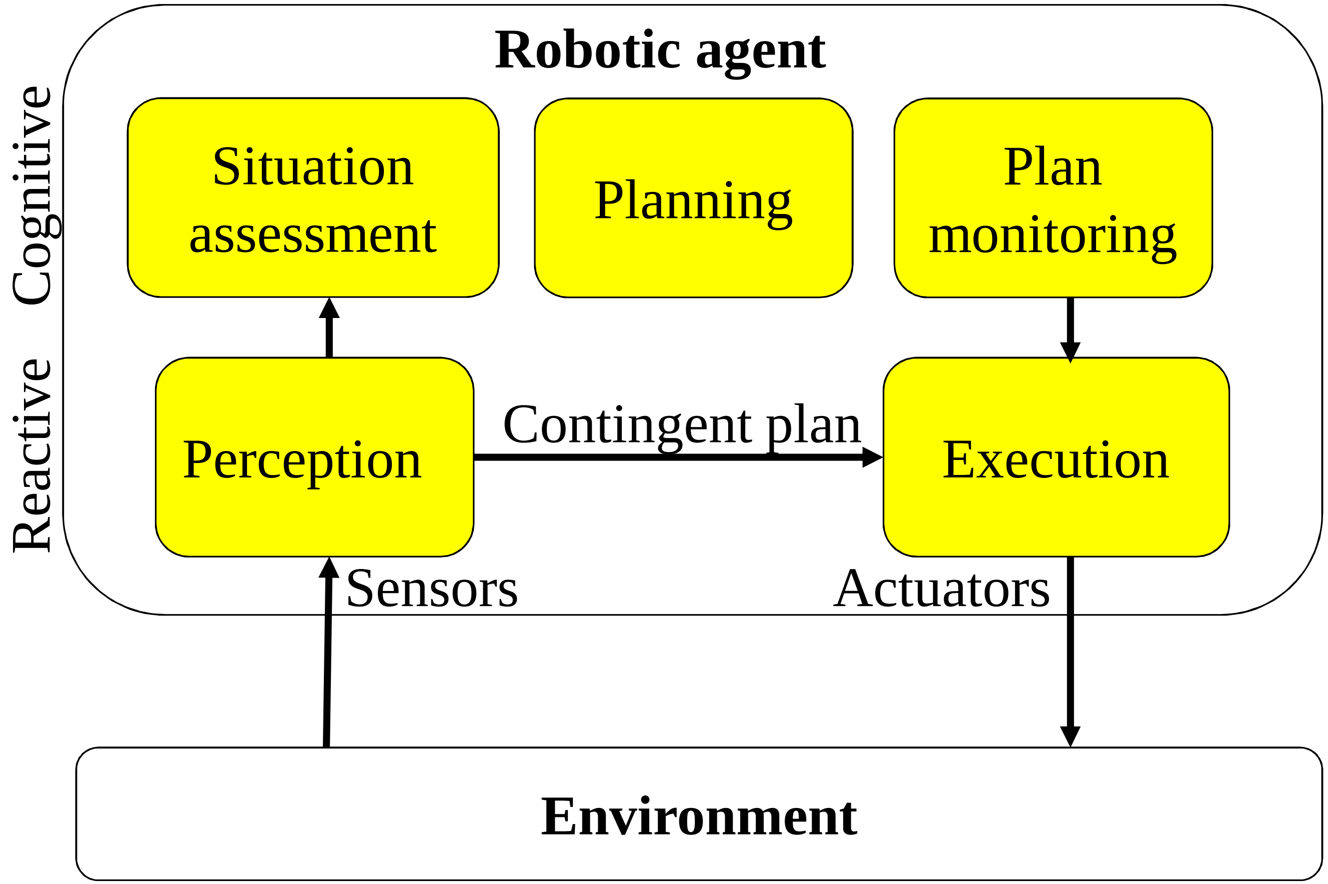
Software architecture: Sense-Plan-Act [Nilsson 80]



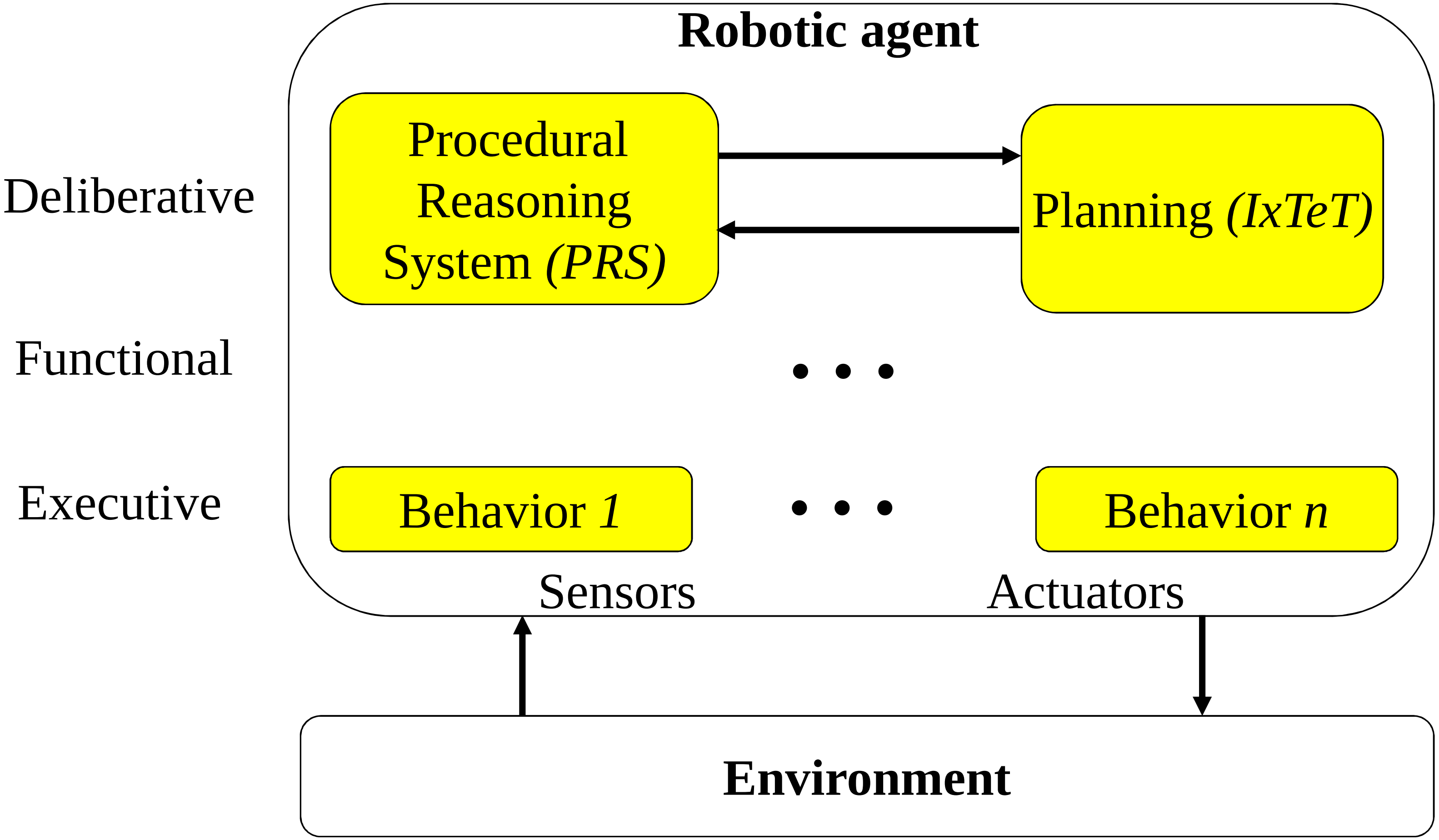
2-level Software Architecture [Hayes-Roth et al. 95]



2-level++ Software Architecture [Baltié et al. 07]



LAAS Software Architecture [Alami et al. 98]



Conclusion

- $S_{out} = S_{in} \cup Post+ \setminus Post-$ is wrong in dynamic multi-agent environments.
- Need for a taxonomy of environments.
- Where do goals come from? Motivations.
- Status of planning:
  - Cognitive psychology (Jean-Michel Hoc).
  - Post hoc rationalization (Lucy Suchman).
  - Opportunistic planning (blackboard, Barbara Hayes-Roth).

References

[Alami et al. 98] R. Alami, R. Chatila, S. Fleury, M. Ghallab, F. Ingrand. *An Architecture for Autonomy*. In *International Journal of Robotics Research* (Special Issue on ``Integrated Architectures for Robot Control and Programming"), Vol 17, N° 4, April 1998. LAAS Report N°97352.

[Baltié et al. 07] J. Baltié, E. Bensana, P. Fabiani, J. – L. Farges, S. Millet, P. Morignot, B. Patin, G. Petitjean, G. Pitois, J. – C. Poncet. *Multi-Vehicle Missions: Architecture and Algorithms for Distributed On Line Planning*. In Dimitri Vrakas and Ioannis Vlahavas (eds.), Artificial Intelligence for Advanced Problem Solving Techniques, Information Science Reference. December 2007.

[Gat 98] Gat, E. *Three-layer architectures*. In D. Kortenkamp et al. Eds. *A.I. and mobile robots*. AAAI Press, 1998.

[Hayes-Roth et al. 95] Hayes-Roth, B.; Pfleger, K.; Morignot, P.; & Lalanda, P. *Plans and Behavior in Intelligent Agents*. Knowledge Systems Laboratory, KSL-95-35, Stanford Univ., CA, March, 1995.

[Nilsson 80] Nils J. Nilsson. *Principles of Artificial Intelligence*. Palo Alto, CA, Tioga. 1980.

[Wolfe et al. 10] J. Wolfe, B. Marthi, S. Russell. *Combining Task and Motion Planning for Mobile Manipulation*. In Proceedings of the International Conference on Automated Planning and Scheduling, Toronto, Canada, 2010.

Thank you for your attention!