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. »

\[
\begin{align*}
\text{State} & : f_1, f_2, \ldots, f_n \\
\text{Goals} & : g_1, g_2, \ldots, g_l
\end{align*}
\]

• 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))))

- Qualification problem; Ramification problem.
- Frame problem: Closed world assumption.

April 29, 2016
Seminar VEH08, VEDECOM
(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) (HANDEMPLOYEE))
   (: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)

with operator:

<table>
<thead>
<tr>
<th>puton ?b ?u ?t</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>(clear ?b)</td>
<td>(not (on ?b ?u))</td>
</tr>
<tr>
<td>(on ?b ?u)</td>
<td>(=&gt; (&lt;&gt; ?u table) (clear ?u))</td>
</tr>
<tr>
<td>(clear ?t)</td>
<td>(on ?b ?t)</td>
</tr>
<tr>
<td></td>
<td>(=&gt; (&lt;&gt; ?t table) (not (clear ?t))))</td>
</tr>
</tbody>
</table>

April 29, 2016
Seminar VEH08, VEDECOM
P. Morignot – 06/01/2012
The anomaly of Gerald Jay Sussman (2 / 2)

Initial
(clear C)
(on C A)
(on A ta.)
(clear B)
(on B ta.)
(clear ta.)

put on C A ?t
(clear C)
(on C A)
(clear ?t)
(not (on C A))
(clear A)
(on C ?t)
(not (clear ?t))

put on B table C
(clear B)
(on B ta.)
(clear C)
(not (on B ta.))
(on B C)
(not (clear B))

put on A table B
(clear A)
(on A ta.)
(clear B)
(not (on A ta.))
(on A B)
(not (clear B))

Final
(on A B)
(on B C)

April 29, 2016
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

Initial → Buy ground → Build house → Final

Initial → Buy ground → Obtain construction license → Build Walls/roof → Pay workers → Final

Initial → Make loan
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:
   \[ initial\_state \land all\_plans\_n \land goals \]
3. Run a SAT solver
4. IF solution found THEN decode \hspace{1cm} // SUCCESS
5. Increment $n$

• Improvement: Try plan lengths in parallel.
Planner using SAT solvers: Encoding

- **Goals:** \( \text{on}(A,B)@T \land \text{on}(B,C)@T \)
- **Initial state:** \( \text{clear}(C)@0 \land \text{on}(C,A)@0 \land \text{clear}(B)@0 \)
  
  \( (\land \neg \text{on}(A,C)@0 \land \neg \text{on}(A,B)@0 \land \neg \text{on}(B,C)@0 \land \neg \text{on}(B,A)@0 \land \neg \text{clear}(A)@0) // \text{closed world assumption} \)

- **Axiom schemas on preconditions:**
  \( \forall \, x, \forall \, y, \forall \, z, \forall \, t : \)
  \( \text{puton}(x, y, z)@t \Rightarrow \text{on}(x,y)@t \land \text{clear}(x)@t \land \text{clear}(z)@t \)

- **Axiom schemas on effects:**
  \( \forall \, x, \forall \, y, \forall \, z, \forall \, t : \)
  \( \text{on}(x,y)@t \land \text{clear}(x)@t \land \text{clear}(z)@t \land \text{puton}(x,y,z)@t \Rightarrow \text{clear}(y)@t+1 \land \text{on}(x,z)@t+1 \)

- **Axiom schemas for one operator at a time:**
  \( \forall \, x, \forall \, y, \forall \, y', \forall \, z, \forall \, z', \forall \, t, y <> y' \land z <> z' : \)
  \( \neg ( \text{puton}(x, y, z)@t \land \text{puton}(x,y',z')@t ) \)

- **Frame axiom schemas:**
  \( \forall \, p, \forall \, t: \)
  \( p@(t+1) \Rightarrow ( p@t \lor a_1^p@t \lor ... \lor a_n^p@t ) \)
  \( \neg p@(t+1) \Rightarrow ( \neg p@t \lor a_1 \neg p@t \lor ... \lor a_n \neg p@t ) \)
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

- 1977: NOAH by Earl Sacerdoti
- 1981: MOLGEN by Mark Stefik
- 1986: IxTeT by Malik Ghallab.
- 1986: SIPE by David Wilkins.
- 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


• PDDL 3.1.: see Wikipedia.

• Conferences:
  • International Conference on Automated Planning and Scheduling (ICAPS). [http://www.icaps.org](http://www.icaps.org)

• Journals:
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 \([ A_1 ; A_2 ; \ldots ; A_n ]\) is a tree composed of steps
  \[
  \text{IF } \langle \text{test} \rangle ~ \text{THEN } <\text{plan-T}> ~ \text{ELSE } <\text{plan-F}>
  \]
• **Example:** 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.
  • \(\text{Action}(\text{Left}, \text{Pre}: \text{AtRight}, \text{Post}: \text{AtLeft} \lor (\text{AtLeft} \land \text{IF LeftClean THEN } \neg \text{LeftClean}))\)
Conditionnal Planning (2 / 3)
Conditionnal Planning (3 / 3)
Online Re-planning (1 / 19)

- **Initial state**
  - $f_1$
  - $f_2$
  - ...
  - $f_n$

- **Goals**
  - $g_1$
  - $g_2$
  - ...
  - $g_l$
Online Re-planning (2 / 19)
Online Re-planning (3 / 19)
Online Re-planning (4 / 19)

Now

$A_1 \rightarrow A_2 \rightarrow \ldots \rightarrow A_i \rightarrow A_{i+1} \rightarrow \ldots \rightarrow A_m$
Online Re-planning (5 / 19)

Now

\[ A_1 \rightarrow A_2 \rightarrow \cdots \rightarrow A_i \rightarrow A_{i+1} \rightarrow \cdots \rightarrow A_m \]
Online Re-planning (6 / 19)

Now
Online Re-planning (7 / 19)

Now

$A_1 \rightarrow A_2 \rightarrow \cdots \rightarrow A_i \rightarrow A_{i+1} \rightarrow \cdots \rightarrow A_m$
Online Re-planning (8 / 19)

Now
Online Re-planning (9 / 19)

Now

\[ A_1 \rightarrow A_2 \rightarrow \cdots \rightarrow A_i \rightarrow A_{i+1} \rightarrow \cdots \rightarrow A_m \]
Online Re-planning (10 / 19)
Online Re-planning (11 / 19)

Initial state

\[ f_1 \quad f_2 \quad \ldots \quad f_n \]

\[ A_1 \rightarrow A_2 \rightarrow \cdots \rightarrow A_i \]

Goals

\[ g_1 \quad g_2 \quad \ldots \quad g_l \]
Online Re-planning (12 / 19)

Initial state

$A_1 \xrightarrow{f_1} A_2 \xrightarrow{f_2} \cdots \xrightarrow{f_n}$

New state

$A_1 \xrightarrow{f'_1} A_2 \xrightarrow{f'_2} \cdots \xrightarrow{f'_n}$

Goals

$g_1 \xleftarrow{g_2} \cdots \xleftarrow{g_l}$
Online Re-planning (13 / 19)
Online Re-planning (14 / 19)
Online Re-planning (15 / 19)

\[ A_1 \rightarrow A_2 \rightarrow \cdots \]

\[ A'_1 \rightarrow A'_2 \rightarrow \cdots \]

Now

April 29, 2016
Online Re-planning (16 / 19)

Now

A_1 \rightarrow A_2 \rightarrow \cdots \rightarrow A'_{k}
Online Re-planning (17 / 19)

Now
Online Re-planning (18 / 19)

\[ A_1 \rightarrow A_2 \rightarrow \ldots \rightarrow A_k \]

\[ A'_1 \rightarrow A'_2 \rightarrow \ldots \rightarrow A'_k \]

Now
Online Re-planning (19 / 19)

Now
Limitation of classical planning

One dimension of the problem
Software architecture: Sense-Plan-Act [Nilsson 80]
2-level Software Architecture [Hayes-Roth et al. 95]
2-level++ Software Architecture [Baltié et al. 07]

Robotic agent

- Situation assessment
- Planning
- Plan monitoring

Perception

- Sensors

Execution

- Actuators

Contingent plan

Environment
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


Thank you for your attention!