Task planning & Agent architectures

Philippe Morignot
Outline

➢ Task planning
  ✓ Statement
  ✓ Difficulty
  ✓ An example
  ✓ Some task planners
  ✓ History & conclusion & references

➢ Robotics-oriented agent architectures
  ✓ Statement
  ✓ Difficulty
  ✓ Some architectures
  ✓ Conclusion & References

➢ Conclusion
Part I ---

Task planning
"Given generic operators, a state and goals, find a sequence of instantiated operators, which lead the initial state to a state which includes the goals."

Solution-plan:

- "Task/action planning" / "plan synthesis" / "action plan generation": activity of building a plan.
- "Planner": computer software which solve this problem.
The problem

- The crane domain:
  - 1 crane, $a$ locations, $b$ trucks, $c$ stacks, $d$ containers.

- If $a = 5$, $b = 3$, $c = 3$, $d = 100$, then $\sim 10^{277}$ states.
- Classical planning is in NP.

You cannot enumerate all states!
Assumptions

- **A 1**: *The agent is the sole cause of change.*
  - No other agent, artificial or human.

- **A 2**: The environment is totally observable, the agent has perfect knowledge of the environment.
  - The agent cannot reason (e.g., plan) on things he does not know.

- **A 3**: *The environment is static.*
  - The environment does not change spontaneously.
Planning Domain Definition Language (PDDL)

- Representation language to define:
  - a domain: operators
  - a problem: initial state and goals.

- An operator is composed of:
  - Pre-condition: term which must hold for the action to be executable.
  - Effect/post-condition: term the truth value of which is changed by the action, when compared to the incoming situation.

- A term might be sometimes true, sometimes false, depending on the time at which it is considered.
  - Logical operator « not »
    - Example: (not (ON MOUSE PAD))
  - « Fluent » (term)
    - Example: (ON MOUSE PAD)
Definitions

- **State**: a symbolic description of the agent / environment.
  - Changes over time, because of the agent actions.
- **Goal**: term which has to be satisfied. (= pre-condition).
  - Might be conjunctive, e.g.: Rich && Handsome && Famous.
- **Sub-goal**: goal obtained by regressing another goal. (= pre-condition).
- **Support**: post-conditions which unify with a given posterior pre-condition.
- **Causal link**: relation between a post-condition and a posterior pre-condition, which is satisfied by this post-condition.
  - E.g.: operator WIN-LOTO et goal (OWN $1,000,000,000)
- **Mutual exclusion** *(mutex)*: 2 pre- or post-conditions which conflict together.
  - E.g.: pre-condition (ON MOUSE PAD) and the post-condition (not (ON MOUSE PAD)) in parallel.
Example of PDDL operator: the blocks world

- Operator:

```
(: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)))
```

- What about the table? And the arm? What if several arms? What about colored blocks? Or with a notch? Or of different sizes?

- Conditionals? Universal quantification?
Example: a planning problem in PDDL

(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) (HANDEMPY))
  (: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 (1/16)

with:

<table>
<thead>
<tr>
<th>puton ?b ?u ?t</th>
</tr>
</thead>
<tbody>
<tr>
<td>(clear ?b)</td>
</tr>
<tr>
<td>(on ?b ?u)</td>
</tr>
<tr>
<td>(clear ?t)</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
The anomaly of Gerald Jay Sussman (2/16)

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

Final
(on A B)
(on B C)
The anomaly of Gerald Jay Sussman (2/16)

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

Final
- (on A B)
- (on B C)
The anomaly of Gerald Jay Sussman (3/16)

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

Final
(on A B)
(on B C)
The anomaly of Gerald Jay Sussman (4/16)

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

Puton A ?u B
- (clear A)
- (on A ?u)
- (clear B)
- (on A B)
- (not (clear B))

Puton B ?u C
- (clear B)
- (on B ?u)
- (clear C)
- (on B C)
- (not (clear C))

Final
- (on A B)
- (on B C)

Initial Final
The anomaly of Gerald Jay Sussman (5/16)

<table>
<thead>
<tr>
<th>Initial</th>
<th>Final</th>
</tr>
</thead>
<tbody>
<tr>
<td>(clear C)</td>
<td>(on A B)</td>
</tr>
<tr>
<td>(on C A)</td>
<td>(on B C)</td>
</tr>
<tr>
<td>(on A ta.)</td>
<td>(clear ta.)</td>
</tr>
<tr>
<td>(clear B)</td>
<td>(clear ta.)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>puton A ?u B</th>
<th>puton B ?u C</th>
</tr>
</thead>
<tbody>
<tr>
<td>(clear A)</td>
<td>(clear B)</td>
</tr>
<tr>
<td>(on A ?u)</td>
<td>(on B ?u)</td>
</tr>
<tr>
<td>(clear ?u)</td>
<td>(clear ?u)</td>
</tr>
<tr>
<td>(on A B)</td>
<td>(on C A)</td>
</tr>
<tr>
<td>(not (clear B))</td>
<td>(not (clear C))</td>
</tr>
</tbody>
</table>
The anomaly of Gerald Jay Sussman (6/16)

Initial

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

puton B ?u C

- (clear B)
- (on B ?u)
- (clear C)
- (on B C)
- (not (clear C))

puton A ?u B

- (clear A)
- (on A ?u)
- (clear ?u)
- (on A B)
- (not (clear B))

Final

- (on A B)
- (on B C)
The anomaly of Gerald Jay Sussman (7/16)

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

| puton B ?u C
| (clear B)
| (on B ?u)
| (clear C)

| puton A ?u B
| (clear A)
| (on A ?u)
| (clear C)

| Final
| (on A B)
| (on B C)
The anomaly of Gerald Jay Sussman (8/16)

Initial

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

Put on B table C

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

Put on A table B

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

Final

(on A B)
(on B C)
The anomaly of Gerald Jay Sussman (11/16)

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

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

Final
(on A B)
(on B C)

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

put on ?b A ?t
(clear ?b)
(on ?b A)
(clear ?t)
(not (on ?b A))
(clear A)
(on ?b ?t)
(not (clear ?t))
The anomaly of Gerald Jay Sussman (12/16)

Initial

- (clear C)
- (on C A)
- (on A ta.)
- (clear B)
- (on B ta.)
- (clear ?t)

Initial

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

Initial

- (clear B)
- (on B ta.)
- (clear C)
- (not (on B ta.))
- (on B C)
- (not (clear C))

puton C A ?t

Initial

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

puton A table B

Final

- (on A B)
- (on B C)

Final

- (clear A)
- (on A ta.)
- (clear B)
- (not (on A ta.))
- (on A B)
- (not (clear B))
The anomaly of Gerald Jay Sussman (13/16)

Initial
- (clear C)
- (on C A)
- (on A ta.)
- (clear B)
- (on B ta.)
- (clear ?t)

Final
- (on A B)
- (on B C)

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

Put on B table C
- (clear B)
- (on B ta.)
- (clear C)
- (not (on B ta.))
- (on B C)
- (not (clear B))
- (not (clear C))
The anomaly of Gerald Jay Sussman (14/16)

<table>
<thead>
<tr>
<th>Initial</th>
<th>puton C A ?t</th>
<th>puton B table C</th>
<th>puton A table B</th>
<th>Final</th>
</tr>
</thead>
<tbody>
<tr>
<td>(clear C)</td>
<td>(clear C)</td>
<td>(clear B)</td>
<td>(clear A)</td>
<td>(on A B)</td>
</tr>
<tr>
<td>(on C A)</td>
<td>(not (on C A))</td>
<td>(not (on B ta.))</td>
<td>(on A ta.))</td>
<td>(on B C)</td>
</tr>
<tr>
<td>(clear B)</td>
<td>(clear A)</td>
<td>(on B C)</td>
<td>(on A B)</td>
<td>(not (clear B))</td>
</tr>
<tr>
<td>(on B ta.)</td>
<td>(on C ?t)</td>
<td>(clear C)</td>
<td>(clear C)</td>
<td>(not (on A ta.))</td>
</tr>
<tr>
<td>(clear ta.)</td>
<td>(not (clear ?t))</td>
<td>(not (clear C))</td>
<td>(not (clear B))</td>
<td>(on B C)</td>
</tr>
</tbody>
</table>
The anomaly of Gerald Jay Sussman (15/16)

<table>
<thead>
<tr>
<th>Initial</th>
<th>puton C A table</th>
<th>puton B table C</th>
<th>puton A table B</th>
<th>Final</th>
</tr>
</thead>
<tbody>
<tr>
<td>(clear C)</td>
<td>(clear C)</td>
<td>(clear B)</td>
<td>(clear A)</td>
<td>(on A B)</td>
</tr>
<tr>
<td>(on C A)</td>
<td>(on C A)</td>
<td>(on B C)</td>
<td>(on B ta.)</td>
<td>(on B C)</td>
</tr>
<tr>
<td>(clear B)</td>
<td>(clear ta.)</td>
<td>(clear C)</td>
<td>(clear C)</td>
<td>(not (clear B))</td>
</tr>
<tr>
<td>(on B ta.)</td>
<td>(not (on C A))</td>
<td>(not (on B ta.))</td>
<td>(not (clear C))</td>
<td>(not (on A ta.))</td>
</tr>
<tr>
<td>(clear ta.)</td>
<td>(not (clear ta.))</td>
<td>(clear ta.)</td>
<td>(not (clear B))</td>
<td>(on B C)</td>
</tr>
</tbody>
</table>
The anomaly of Gerald Jay Sussman (16/16)

<table>
<thead>
<tr>
<th>Initial</th>
<th>puton C A table</th>
<th>puton B table C</th>
<th>puton A table B</th>
<th>Final</th>
</tr>
</thead>
<tbody>
<tr>
<td>(clear C)</td>
<td>(clear C)</td>
<td>(clear B)</td>
<td>(clear A)</td>
<td>(on A B)</td>
</tr>
<tr>
<td>(on C A)</td>
<td>(not (on C A))</td>
<td>(not (on B ta.))</td>
<td>(on A ta.)</td>
<td>(on B C)</td>
</tr>
<tr>
<td>(clear B)</td>
<td>(on B ta.)</td>
<td>(on B C)</td>
<td>(clear C)</td>
<td>(on A B)</td>
</tr>
<tr>
<td>(on B ta.)</td>
<td>(clear C)</td>
<td>(not (clear C))</td>
<td>(clear B)</td>
<td>(not (clear B))</td>
</tr>
<tr>
<td>(clear ta.)</td>
<td>(on C ta.)</td>
<td>(on C ta.)</td>
<td>(on C ta.)</td>
<td>(on B C)</td>
</tr>
</tbody>
</table>
The anomaly of Gerald Jay Sussman: solution

(1) (2)

(3) (4)
Plan = (Templates T, Operators Op, PartialOrder O, Unification U).

Definitions:

✓ Conflict: a post-condition might destroy a causal link (threat).

✓ Satisfying a pre-condition p: adding an operator before p, which a post-condition q unifies with p (i.e., q = p).

PLANNER1(T, Op., O, U):

WHILE(conflict || at least 1 unsatisfied pre-condition)

1. Solve conflicts:
   • Add a unification / non unification constraint;
   • Add a precedence constraint.

2. Choose an unsatisfied pre-condition p;

3. Satisfy p:
   • Add a unification / non-unification constraint;
   • Add a precedence constraint;
   • Add an operator.

END WHILE
Algorithms (2/3) : in the state space

- A state $S_i$ = a state of the environment.
  - $S_0$ = initial state, composed of the fluents $f_1, \ldots, f_n$.
  - Example: a given configuration of blocks.
- $\text{Successors}(S_i) = n$ states $S_{i+1}, \ldots, S_{i+n}$ which can be reached by an instantiated operator applicable in $S_i$.
- $\text{Solution}(S_i)$ iff $S_i$ includes the goals $g_1, \ldots, g_l$.

$\text{PLANNER2}(\text{Alg.}, S_0, \text{Successors}, \text{Solution})$ :
  - Alg. = any state-space search algorithm.
  - Heuristics ?

Example:
  - Algorithm A* of Heuristic Search Planner (HSP) [Bonet 98]
  - Heuristics: a graph plan without negative pre-conditions.
Algorithms (3/3) : in the plan space

- A state = a partially-ordered partially instantiated plan.

- **PLANNER3(T, Op., O, U)** :
  
  Search algorithm in a space of states (e.g., A*) :
  
  - A state = a partial plan \((T, Op., O', U')\).
  - A successor = obtained by solving a conflict, or satisfying a pre-condition.
    - Adding a unification / non-unification constraint to \(U\)
    - Adding a precedence constraint to \(O\)
    - Adding an operator from \(T\) to \(Op\)
  - A solution function = no conflict && all pre-conditions are satisfied.

- Heuristics?

- Example:
  
  ✓ Universally quantified Conditional Partial-Order Planner (UCPOP) [Penberthy 92]
Plan of graph [Blum 97]

Levels include mutual exclusions (mutex): a level is not a state!
Forward development of a plan of graph, backward search.
Hierachical Tasks Network (HTN)

- **Additional knowledge:**
  - A task can be decomposed into sub-tasks.

  ![Hierarchical Tasks Network Diagram]

- **Search algorithm by refining plans.**
  - Simple Task Network (STN).

- **Example:**
  - The HTN planner used by Jason Wolfe at Willow Garage (CA).
  - SHOP and SHOP2, by Dana Nau from Univ. Maryland.
SAT-based planning [Kautz 92]

The SAT problem:

✓ Find a truth value for each proposition, which together satisfy a given logical formula, using AND, OR and NOT.
✓ Proposition logics.
✓ 1st problem which has been proved NP-complete (1971).

Principle of SAT-based planning:

1. \( n = 1 \) // Length of the solution-plan.
2. Turn a plan of length \( n \) into a formula in proposition logic.
3. Attempt to prove this formula using a SAT solver.
4. IF it fails, THEN (i) increment \( n \), (ii) GOTO 2.

Example of formulas, in the blocks world:

✓ For all \( x, y, z, i \):
  - \( \text{on}(x, y, i) \land \text{clear}(x, i) \land \text{clear}(z, i) \land \text{puton}(x, y, z, i) \Rightarrow \text{clear}(y, i+1) \land \text{on}(x, z, i+1) \)

✓ For all \( x, x', y, y', z, z', i \):
  - \( x \neq x' \land y \neq y' \land z \neq z' \Rightarrow \neg \text{puton}(x, y, z, i) \lor \neg \text{puton}(x', y', z', i) \)
CSP-based planning

- **Constraint programming:**
  - **Statement:** for each variable, search for a value from the variable’s domain, so that all values satisfy the constraints.
  - **Representation:** Variables / Domains / Constraints.
  - **Algorithm:**
    - Choose a variable
    - Choose a value from the variable’s domain
    - Propagate this assignment through constraints
    - If a domain becomes empty, backtrack on previous choices.

- **Principle of CSP-based planning:**
  1. Heuristically estimate the length \( n \) of a solution-plan;
  2. Turn the planning problem into a dynamic CSP;
  3. Attempt at finding a solution plan of length \( n \) using a CSP solver;
  4. IF failure, THEN increment \( n \) ; GOTO 2.

- **Examples :**
  - IxTeT planner from LAAS-CNRS in Toulouse [Laborie 95].
  - Constraint Programming Temporal planner (CPT) from ONERA Toulouse [Vidal 06].
    - [http://v.vidal.free.fr/oneraq#cpt](http://v.vidal.free.fr/oneraq#cpt)
Conditional planning (1 / 2)

- The agent may not know the output of its own actions.
- Plans have branches:
  - IF \(< test >\) THEN $Plan_A$ ELSE $Plan_B$
  - Obtain a plan in every case
- **Full observability**: the agent knows its state.
  - No need for an operator « observe ».
- Actions might fail: disjunctive effects.
- Conditional effects.
  - EFFECTS : IF To_Left, CleanL ; IF To_Right, CleanD.
- Conditional planning is harder than NP.
- **Example (double Murphy)**: a vacuum-cleaner agent must clean all rooms.
  - Rule 1: the vacuum cleaner sometimes drops dust when it moves to a clean room.
  - Rule 2: the vacuum cleaner sometimes drops dust if CLEAN is executed in a clean room.
Conditional planning (2 / 2)

➢ Search in an AND-OR graph:
History

1971: STRIPS from Richard Fikes.
1977: NOAH from Earl Sacerdoti
1981: MOLGEN from Mark Stefik
1986: IxTeT from Malik Ghallab.
1986: SIPE from David Wilkins.
1997: GRAPHPLAN from Avrim Blum & Merrick Furst.
2000: HSP from Hector Geffner.
2000: YAHSP from Vincent Vidal.
2001: FF from Jörg Hoffmann,
2005: CPT from Vincent Vidal.
2007: DAE from Marc Schoenauer.
References (1 / 2) – Part I


**Conférences :**

- International Conference on Automated Planning and Scheduling (ICAPS). http://www.icaps.org

**Journals :**

References (2 / 2) – Part II

Conclusion of part I (1/2)

- Domain explored for 40 years
- Some planners now are available:
  - CPT, FF, SATPLAN, ...
  - International Planning Competition (IPC).
- Conditional planning: more difficult…
- Properties of a planner:
  - Correctness
  - Completeness
  - Optimality
  - Canonicity
  - Efficiency

- Hint:
  - Merge probabilistic planning (MDP) and symbolic planning (STRIPS).
Demo of the CPT task planner...
Planning & Execution [Russel 2010]

When to plan?

- Before executing (off-line planning).

- While executing (on-line planning).
Part II ---

Robotics-oriented agent architectures
« An agent is a system including reasoning (e.g., temporal), perceiving its environment, acting on it and interacting with other agents (artificial or human). »
Difficulty

One dimension of the problem

Response time

- Deliberation
- Reaction
Examples of robotic agents
Architecture Sense-Plan-Act [Nilsson 80]

Robotic agent

Perception → Planning → Execution

Sensors → Actuators

Environment
Subsumption architecture [Brooks 85]

- No symbol [Brooks 91].

Diagram:

- Robotic agent
  - Finite-state automaton $n$
  - $\cdots$
  - Finite-state automaton 2
  - Finite-state automaton 1

- Environment

- Parameters
  - Sensors
  - Actuators

[Image of the diagram]
2-level architecture [Hayes-Roth et al. 95] (1/3)
2-level architecture [Hayes-Roth et al. 95] (2/3)

Agent

Level...

Behavior 1  Behavior 2  ...  Behavior N

Controller

...
2-level architecture [Hayes-Roth et al. 95] (3/3)

- **Decisional level**
  - **Events**
  - **Action descriptions**

- **Reactive level**
  - **Sensors**
  - **Actuators**

**Agent**

- **Environment**

- **Reaction time**
  - 10 s
  - 1 s
  - 0.1 s
2-level++ architecture [Baltié et al. 07]

Robotic agent

Cognitive

- Situation assessment
- Planning
- Plan monitoring

Reactive

- Perception
- Contingent plan
- Action

Sensors Actuators

Environment
3-level architecture [Gat 98]

Deliberator
- Algo. 1
- Algo. m

Sequencer

Controller
- Behavior 1
- Behavior n

Robotic agent

Sensors

Actuators

Environment
LAAS-CNRS architecture [Alami et al. 98]

Robotic agent

Deliberative
Procedural Reasoning System (PRS)

Action planning (IxTeT)

Functional

Executive

Behavior 1

... ...

Behavior n

Sensors

Actuators

Environment
Architectures for robots

- Open RObotic COntrol Software, from H. Bruyninckx (Belgium).
  - Ontology, no task planning.

- Robotic Operating System (ROS), from Willow Garage (CA).
  - An HTN planner [Wolfe 10], no ontology.
An ontology for the robot SAM

Context analysis
A, B, C, D

Contextual information (relations, states)
Before(A, B)
Clear(C)
OutOfRange(D)

Determination method

Ontology

Type A
Type B
Type C
before
Stacked on

Determination method
Conclusion of Part II

- Domain explored since [Nilsson 80], i.e., ~30 years.
- No unique architecture makes consensus!
- Critical properties:
  - **Real-time**
    - How to get immediately a good reaction?
  - **Safety**
    - How to get a good reaction in the worst case?
- Hints:
  - Ontology to analyze the context.
  - Multi-Agent Systems: *intelligence emerges from interaction among agents.*
References (1 / 2) – Part II


References (2 / 2) – Part II

General conclusion

\[ f(x) = e^x \]

If P is different than NP, then we are fighting against the exponential function in the worst case!
Thank you for your attention!