Wrocław
University
of Science
and Technology

Mobile robotics
Introduction to task planning

Janusz Jakubiak

Department of Cybernetics and Robotics

2023/2024

HR EXCELLENCE IN RESEARCH

Wrocław University of Science and Technology

Copyright information

Following slides are a supporting material for the course Mobile robotics. Most of included notes and illustrations are copyrighted by their authors or publishers. Please respect that copyright by using the notes only for purposes of assigned reading in this class.

Task planning

Examples

- How to cross an intersection?
- How to determine the path in a maze?
- How to build a map of unknown terrain in the fastest way?
- How to deliver objects to the destination points?

Task planning – house robot example

Task

We want to give the robot a high-level task that the robot should carry out automatically using basic actions

Task planning – house robot example

Task

We want to give the robot a high-level task that the robot should carry out automatically using basic actions

Example: the task of placing objects

- Inform the robot where the objects are located (initial state)
- Inform the robot where they are to be located (final state)
- The robot selects the sequence of actions on its own

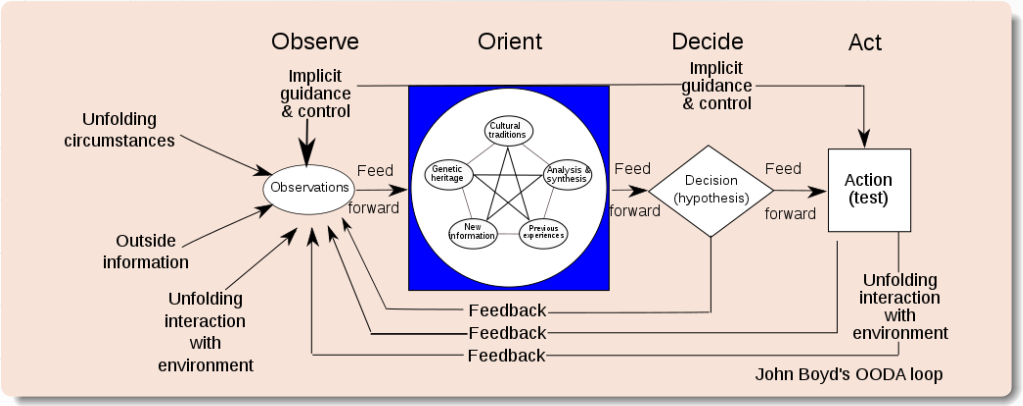
„blocks world”

Task planning

Action scheme

- Collect information
- Determine current and target state
- Analyze possible solutions
- Make a decision

OODA (Observe, Orient, Decide, Act – John Boyd)



Orienting to the environment

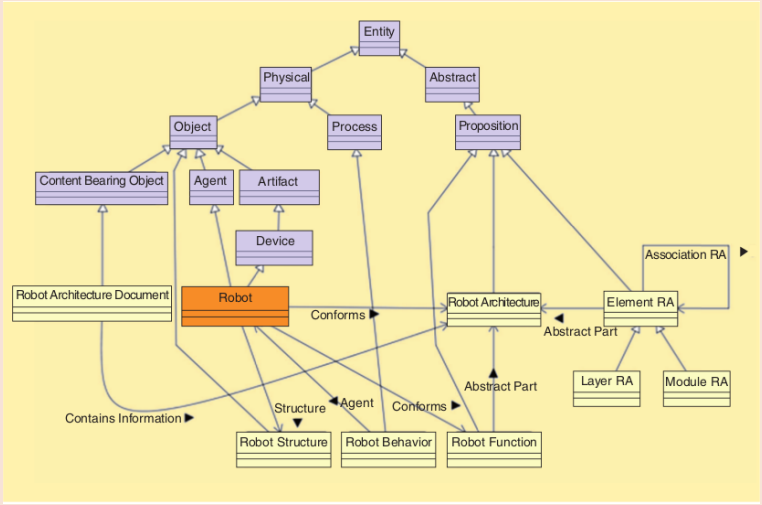
A three-stage cognitive process

1. Perception
   - in: information from the environment
   - out: focusing attention on a phenomenon
2. Interpretation
   - in: transformation of data into a model of the situation
   - out: transformation of operational objectives into actions
3. Reasoning

   matching situation model and behavior patterns

situational awareness

Ontology



1872-2015 – IEEE Standard; S.R. Fiorini et al. A Suite of Ontologies for Robotics and Automation
10.1109/MRA.2016.2645444

Representation of states and actions in propositional logic

State

Relational description of the interrelationships between objects, e.g. `on(obX,obY)` `beside(obX,obY)`

Actions

- describes the transition between two states
- defines the action of an action

Representation of states and actions in propositional logic

State

Relational description of the interrelationships between objects, e.g. `on(obX,obY)` `beside(obX,obY)`

Actions

- describes the transition between two states
- defines the action of an action

Action definition:

```
name(parameters)
PRECOND: initial conditions
EFFECTS: conditions after the action is executed
```

propositional logic otherwise *propositional calculus*,*statement logic*, *sentential calculus*, *sentential logic*

Example – AGV robot

```
objects: robot storage machine component
relations: beside(X,Y) on(X,Y)
actions:
 get(Z,X):
  PRECOND: on(Z,X), not on(Z,robot), beside(X,robot)
  EFFECTS: not on(Z,X), on(Z,robot)

 place(Z,X):
  PRECOND: on(Z,robot), not on(Z,X), beside(X,robot)
  EFFECTS: not on(Z,robot), on(Z,X)

 goto(X):
  PRECOND:
  EFFECTS: beside(X,robot)
```

Example – AGV robot

```
objects: robot storage machine component
relations: beside(X,Y) on(X,Y)
actions:
 get(Z,X):
  PRECOND: on(Z,X), not on(Z,robot), beside(X,robot)
  EFFECTS: not on(Z,X), on(Z,robot)

 place(Z,X):
  PRECOND: on(Z,robot), not on(Z,X), beside(X,robot)
  EFFECTS: not on(Z,robot), on(Z,X)

 goto(X):
  PRECOND:
  EFFECTS: beside(X,robot)
```

initial state: `on(component,storage), not on(component,machine)`

goal: `on(component,machine)`

What series of actions solves the task?

Intelligent systems

- Are Capable of performing useful functions based on a defined purpose and knowledge of the current state
- Mimic biological cognitive processes
- Process information to achieve a goal
- Can learn from examples
- Adapt to changing environmental conditions

Task planning in multi-robot systems

Examples of (additional) goals

- Avoiding interference with the actions of other agents
- Coordination of actions
- Competition with other agents

Wrocław University of Science and Technology

Task planning in multi-robot systems

Examples of (additional) goals

- Avoiding interference with the actions of other agents
- Coordination of actions
- Competition with other agents

Functions

- Choosing between individual and group goals
- Optimization of resource allocation
- Conflict resolution

Wrocław University of Science and Technology

Action planning languages – intro

Key assumption: paradigm shift

Instead of describing sequences of events or predefined strategies, defining the available actions and goals

Action planning languages – intro

Key assumption: paradigm shift

Instead of describing sequences of events or predefined strategies, defining the available actions and goals

Goal Oriented Action Planning (GOAP)

GOAP properties

- \+ large spectrum of possible behavior sequences that can be generated
- \+ ease of managing behavior rules
- \+ generation of different behaviors when requirements change without changing the sequence by the user
- \- the need to solve complex planning problems (often in real time)

STRIPS

1971 R.E. Fikes N.J. Nilsson

Planning problem description

1. initial state s_0
2. conditions fulfilled by the goal G
3. available actions

STRIPS

1971 R.E. Fikes N.J. Nilsson

Planning problem description

1. initial state s_0
2. conditions fulfilled by the goal G
3. available actions

Planning algorithm

- The output is a sequence of actions starting in the initial state and leading to a state satisfying the conditions of the goal
- The goal of the algorithm is to automatically find a solution

Wrocław University of Science and Technology

STRIPS – a task

Initial state

1. representation of properties in the form of first-order logic predicates
2. closed-world assumption (any undefined sentence treated as false)

STRIPS – a task

Initial state

1. representation of properties in the form of first-order logic predicates
2. closed-world assumption (any undefined sentence treated as false)

Goal

1. property representation in the form of first-order logic predicates
2. can be partially defined (i.e., without closed world assumption: state s satisfies G if all literals of G are contained in s)

STRIPS – actions

Initial conditions

1. literals describing the conditions that must be met to be able to activate the action

Result

1. literals describing the state after the execution of the action

Advantages of STRIPS

1. simple formalism
2. easy to check which actions can be activated in a given state
3. easy determination of the state after the execution of the action
4. easy verification of the achieved goal
5. planning e.g. A* (but determining the heuristic function is not trivial)
6. solution "forward"(from s_0) or "backward"(from G), algorithms independent of the domain

Other approaches: ADL, action languages, fluent calculus, event calculus

Planning Domain Definition Language (PDDL)

1. Syntax similar to programming languages
2. Enable description by STRIPS, ADL and others
3. Allows the use of universal planners
4. Prefix notation

```
(beside a b)
(not (beside a b))
(and (beside a b) (beside b c))
(beside ?x ?y)
```

PDDL – components

Domain

1. states (:predicates)
2. actions (:action)

PDDL – components

Domain

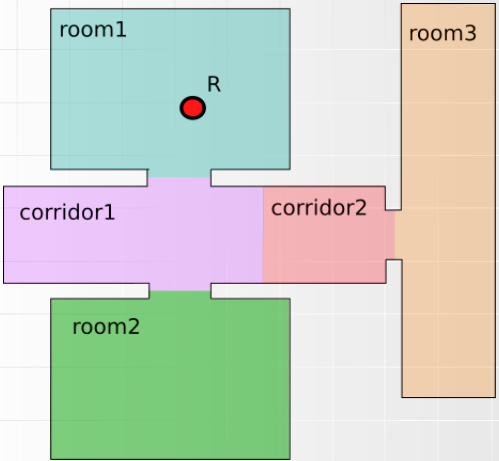
1. states (:predicates)
2. actions (:action)

Task

1. objects (:objects)
2. initial state (:init)
3. objective (:goal)

full list of components in „PDDL - The Planning Domain Definition Language”
https://homepages.inf.ed.ac.uk/mfourman/tools/propplan/pddl.pdf

Example 1 – introduction



Example 1 – domain

Predicates

```
(:predicates
  (beside ?x ?y) (in ?x ?y)
)
```

Example 1 – domain

Predicates

```
(:predicates
  (beside ?x ?y) (in ?x ?y)
)
```

Actions

```
(:action go
 :parameters (?r ?from ?to)
 :precondition (and  (in ?r ?from)
   (or (beside ?from ?to) (beside ?to ?from)) )
 :effect (in ?r ?to)
)
```

Example 1 – task

Objects

```
(:objects R room1 room2 room3 corridor1 corridor2  )
```

Example 1 – task

Objects

```
(:objects R room1 room2 room3 corridor1 corridor2  )
```

Initial state

```
(:init  (in R room1)
 (beside room1 corridor1) (beside room2 corridor1)
 (beside room3 corridor2) (beside corridor1 corridor2)
)
```

Example 1 – task

Objects

```
(:objects R room1 room2 room3 corridor1 corridor2  )
```

Initial state

```
(:init  (in R room1)
 (beside room1 corridor1) (beside room2 corridor1)
 (beside room3 corridor2) (beside corridor1 corridor2)
)
```

Objective

```
(:goal
 (in R room3)
)
```

Example 1 – solution

http://lcas.lincoln.ac.uk/fast-downward/

http://editor.planning.domains/

Example 1 – solution

http://lcas.lincoln.ac.uk/fast-downward/
http://editor.planning.domains/

The result:

```
(go r room1 corridor1)
(go r corridor1 corridor2)
(go r corridor2 room3)
```

Practice

Examplary tasks

1. Describe the task for a more complex floor plan
2. Define the problem of passage for 2 robots
3. Define a problem that takes into account doors (opening and closing)

See also https://github.com/aibasel/pyperplan/tree/master/benchmarks

ADL (action description language)

STRIPS **Language**	ADL **Language**
Only positive literals in states: $Poor \wedge Unknown$	Positive and negative literals in states: $\neg Rich \wedge \neg Famous$
Closed World Assumption: Unmentioned literals are false.	Open World Assumption: Unmentioned literals are unknown.
Effect $P \wedge \neg Q$ means add P and delete Q.	Effect $P \wedge \neg Q$ means add P and $\neg Q$ and delete $\neg P$ and Q.
Only ground literals in goals: $Rich \wedge Famous$	Quantified variables in goals: $\exists x\, At(P_1, x) \wedge At(P_2, x)$ is the goal of having P_1 and P_2 in the same place.
Goals are conjunctions: $Rich \wedge Famous$	Goals allow conjunction and disjunction: $\neg Poor \wedge (Famous \vee Smart)$
Effects are conjunctions.	Conditional effects allowed: **when** P: E means E is an effect only if P is satisfied.
No support for equality.	Equality predicate $(x = y)$ is built in.
No support for types.	Variables can have types, as in $(p : Plane)$.

Figure 11.1 Comparison of STRIPS and ADL languages for representing planning problems. In both cases, goals behave as the preconditions of an action with no parameters.

Other approaches

What has not been taken into account

1. incomplete or ambiguous information
2. non-deterministic actions
3. time of the actions
4. changes in the world not controlled by the agent
5. events corresponding to sensors

Review questions

1. How does task planning differ from motion planning?
2. How are actions defined in the predicate logic?
3. How is description defined in PDDL?