Assignment 1 and PDDL

Michaela Urbanovská

PUI Tutorial
Week 2

Lecture check

- Any questions regarding the lecture?

Assignment 1 - General information

- All details HERE
- **Main task:** implement an optimal planner using A* algorithm and one admissible heuristic
- **Impementation report:** short report to prove that you understand the implementation
- **Heuristics:** LM-Cut, h^{max}
- **Languages:** C, C++, Java
- Automatic evaluation in BRUTE
- **Deadline:** 5.4.2021
- **Questions:** Assignment questions channel + email + forum

Assignment 1 - Data

- All data will be provided to you
- **Test set**
  - limit 10 minutes
- **Bonus data set**
  - 5 bonus data sets in total
  - limit 10 minutes (for each one)
  - +2 points (for each one)

Assignment 1 - Points

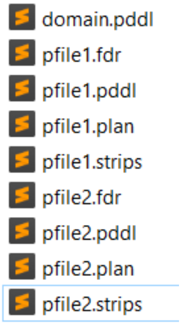
Heuristic	Max points	Lecture
h^{max}	10 + 2*5 (bonus)	3
LM-Cut	20 + 2*5 (bonus)	4

BRUTE will give you **1 point** if the test set is evaluated correctly and full amount of points will be assigned after we check the code and report manually. In case of any problems we will contact you for explanation.

Assignment 1 - Implementation

- Each domain has
  - PDDL domain description
  - Different problem instances

- Each problem instance has
  - STRIPS definition
  - FDR definition
  - PDDL problem description
  - Plan



domain.pddl
pfile1.fdr
pfile1.pddl
pfile1.plan
pfile1.strips
pfile2.fdr
pfile2.pddl
pfile2.plan
pfile2.strips

Assignment 1 - Implementation

Your planner will be called as

```
$ ./planner problem.strips problem.fdr
```

- **Input** in both STRIPS and FDR (you select one to work with)
- **Output** in .plan format on standard output
  - Cost of the plan
  - Heuristic value in initial state
  - Sequence of operators

```
;; Cost: 10
;; Init: 4

(lift hoist0 crate1 pallet0 depot0)
(load hoist0 crate1 truck1 depot0)
(lift hoist1 crate0 pallet1 distributor0)
(drive truck1 depot0 distributor0)
(load hoist1 crate0 truck1 distributor0)
(unload hoist1 crate1 truck1 distributor0)
(drop hoist1 crate1 pallet1 distributor0)
(drive truck1 distributor0 distributor1)
(unload hoist2 crate0 truck1 distributor1)
(drop hoist2 crate0 pallet2 distributor1)
```

Assignment 1 - Implementation

- Plan **validation** is important to see if the planner generated valid plans
- Precompiled validate program at assignment's page
- GitHub repo provided so you can compile it yourself

Validate program is called as

```
$ ./validate -v domain.pddl problem.pddl problem.plan
```

Assignment 1 - Report

- Keep it **short**
- Implementation description / documentation
- Interesting implementation details? Solved issues along the way?
- No need to run into every detail
- 1-3 pages (keep it short, like for real)

PDDL Language

- Planning Domain Definition Language
- General language to describe planning problems
- Syntax similar to LISP, uses predicate logic
- **Domain definition**
  - defines types, predicates, actions (= operators)
- **Problem definition**
  - defines objects, initial state, goal
- Online editor: *http://editor.planning.domains*

PDDL Language

Domain file structure

(define (domain *name*)

PDDL definition of types

PDDL definition of predicates

PDDL definition of actions

)

PDDL Language

Domain file structure

(define (domain *name*)
    PDDL definition of types
    PDDL definition of predicates
    PDDL definition of actions
)

Problem file structure

(define (problem *name*) (:domain *name*)
    PDDL definition of objects
    PDDL definition of initial state
    PDDL definition of goal
)

PDDL Language

Domain file structure

(define (domain *name*)

PDDL definition of types

PDDL definition of predicates

PDDL definition of actions

)

Problem file structure

(define (problem *name*) (:domain *name*)

PDDL definition of objects

PDDL definition of initial state

PDDL definition of goal

)

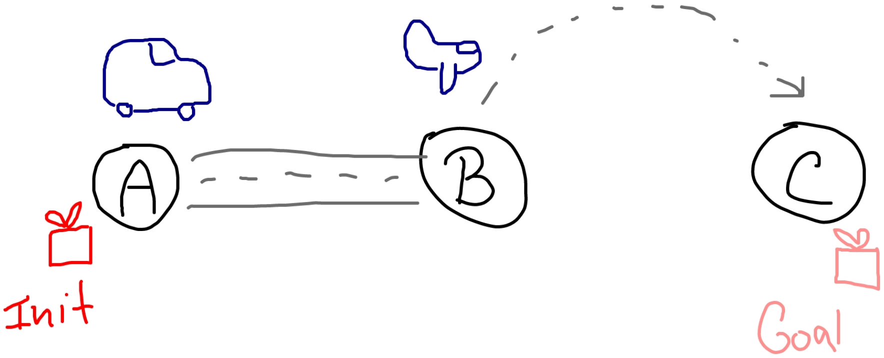
But what does that mean?

PDDL Example

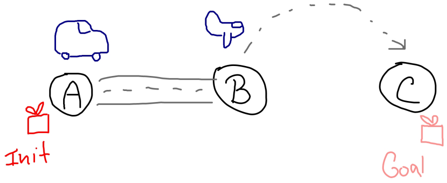
Remember our toy example from the last time?

PDDL Example

Remember our toy example from the last time?



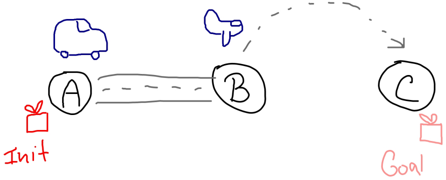
PDDL Example



- Domain definition
  - **Types** - location, vehicle, object, ...
  - **Predicates** - Is truck at A? Is package on the plane? Is plane in C? ...
  - **Actions** - load the package, drive the truck, fly the plane, ...

Let's try it out (http://editor.planning.domains)

PDDL Example

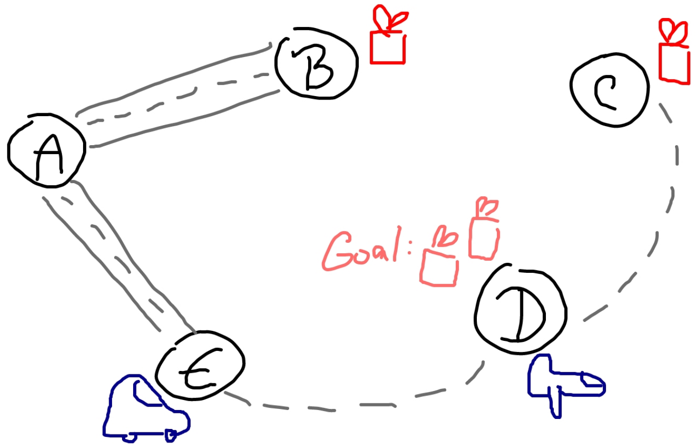


- Problem definition
  - **Objects** - cities, different vehicles, package ...
  - **Initial state** - where do the objects start?
  - **Goal** - what do we want to achieve?

Let's try it out (http://editor.planning.domains)

PDDL Example

Use the existing domain file to define this problem



PDDL Language

- Many other things possible in PDDL
    - negative preconditions
      (not (at ?p ?loc))
    - conditional effects
      (when CONDITION EFFECT)
    - universally quantified formula
      (forall (?a1 - type1 ?a2 - type2 ...) EFFECT)
    - existentially quantified formula
      (exists (?a1 - type1 ?a2 - type2 ...) EFFECT)
    - action costs
      (:functions (total-cost) - number); (increase (total-cost) 5) in effects

Recap

- Assignment 1 explained
- PDDL language
- domain and problem definition
- know how to write it and read it

The End

Feedback form