Lecture 9

CS 6260 Automated Planning and Learning

PDDL 1.2 Features

Department of Computer Science and Engineering

Indian Institute of Technology Madras

Logistics

Announcements

- Office Hours (in SSB 304)
  - Every Tu/We/Th: For one hour after the class starting March 10.
  - Exception Next Week:
    - Office Hours for 2 hours on Tuesday (12:00 pm – 02:00 pm)
    - No class and no office hours on We/Th.

- No classes from Feb 23 – Mar 06.
  - Will post video material to watch.
  - Q/A session and Quick Recap on March 10 (and March 11, if needed).

- Will release Assignment1 and Practice Quiz over the weekend.
  - Quiz solutions will be released on Monday (Feb 15).

Recap: STRIPS

STRIPS-like PDDL

```
(:predicates (on ?x ?y)
             (ontable ?x)
             (clear ?x)
             (handempty)
             (holding ?x)
             )
```

Predicates: Used to define state space of the environment when grounded with objects

```
(:action pick-up
         :parameters (?x)
         :precondition
             (and (clear ?x)
               (ontable ?x)
               (handempty))
         :effect
             (and (not (ontable ?x))
               (not (clear ?x))
               (not (handempty))
               (holding ?x))
)
```

Precondition: This condition must be true for this action to execute

Effect: This is a set of conditions, one of which becomes true when this action is executed
(like add/delete effects in STRIPS)

Requirements

```
(define (domain BLOCKS)
  (:requirements :strips)
  (:predicates (on ?x ?y)
               (ontable ?x)
               (clear ?x)
               (handempty)
               (holding ?x)
               )
```

Requirements??

Features of PDDL supported by this domain file

PDDL 1.2 Features

Refer to this document for PDDL 1.2 details:
https://www.cs.cmu.edu/~mmv/planning/readings/98aips-PDDL.pdf

Additional Info: https://planning.wiki/ref/pddl

:typing

```
(:requirements :strips :typing)
(:predicates (on ?x ?y)
             (ontable ?x)
             (clear ?x)
             (handempty)
             (holding ?x)
             )
```

```
(:requirements :strips :typing)
(:types BLOCKS)
(:predicates
             (on ?x - BLOCKS ?y - BLOCKS)
             (ontable ?x - BLOCKS)
             (clear ?x - BLOCKS)
             (handempty)
             (holding ?x - BLOCKS)
             )
```

:typing

- Also supports hierarchies of objects.

```
(:requirements :strips :typing)
(:types PACK TRUCK DRIVER - OBJ
        LOCATION)
```

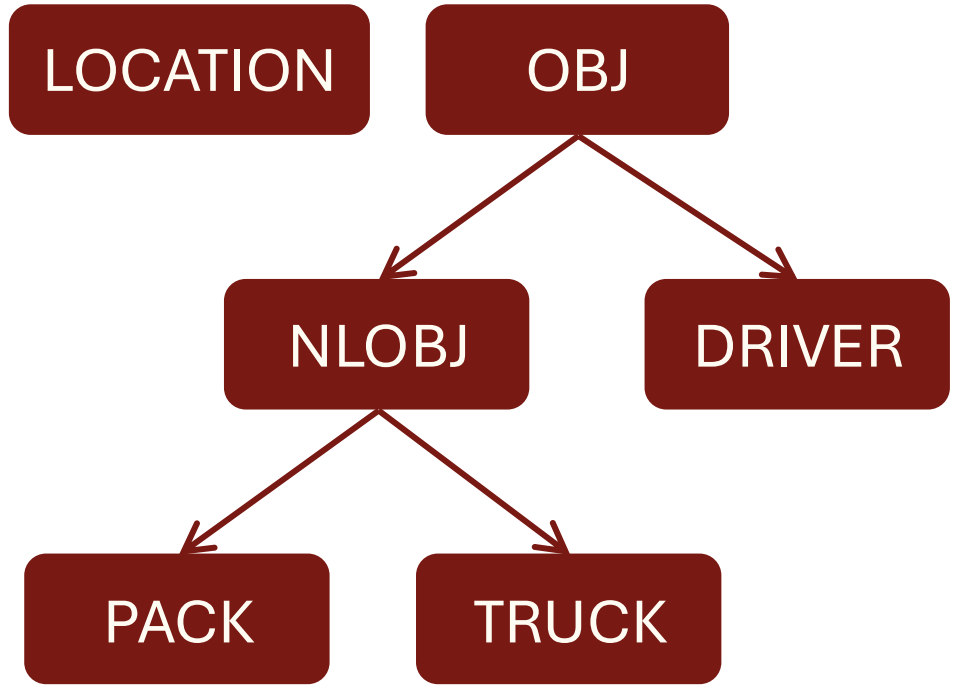
- Advantage of typing?
  - Can define predicates over multiple types.

```
(at ?obj - OBJ ?loc - LOCATION)
```

:typing

- Also supports nested hierarchies of objects.

```
(:requirements :strips :typing)
; NLOBJ stands for "Non-Living Object"
(:types PACK TRUCK - NLOBJ
  NLOBJ DRIVER - OBJ
  LOCATION)
```



:typing

- Hands-on exercise: Convert the driverlog domain from non-typed to typed.

- Access the modified domain and problem here:

https://editor.planning.domains/#read_session=miDI920bVw

:disjunctive-preconditions

```
(:action move
  :parameters
     (?robot ?room1 ?room2 ?door1 ?door2)
  :precondition (and
     (in ?robot ?room1)
     (adjacent ?room1 ?room2)
     (or
        (open ?door1)
        (open ?door2)
     ))
  :effect (and
     (in ?robot ?room2)
     (not (in ?robot ?room1)))
)
```

Equivalent to two different actions
(as we saw earlier)

:equality

```
(:action move
  :parameters
     (?robot ?room1 ?room2 ?door1 ?door2)
  :precondition (and
     (not (= ?door1 ?door2))
     (not (= ?room1 ?room2))
     (in ?robot ?room1)
     (adjacent ?room1 ?room2)
     (or
        (open ?door1)
        (open ?door2)
     ))
  :effect (and
     (in ?robot ?room2)
     (not (in ?robot ?room1)))
)
```

:existential-precondition

```
(:action move
   :parameters
      (?robot ?room1 ?room2 ?door1 ?door2)
   :precondition (and
      (not (= ?door1 ?door2))
      (not (= ?room1 ?room2))
      (in ?robot ?room1)
      (adjacent ?room1 ?room2)
      (or
         (open ?door1)
         (open ?door2)
      ))
   :effect (and
      (in ?robot ?room2)
      (not (in ?robot ?room1)))
)
```

Allows **exists** clause in precondition and goal

:existential-precondition

```
(:action move
  :parameters
     (?robot ?room1 ?room2 ?door1 ?door2)
  :precondition (and
     (in ?robot ?room1)
     (adjacent ?room1 ?room2)
     (exists (?d - DOOR)
          (open ?d)
     ))
  :effect (and
     (in ?robot ?room2)
     (not (in ?robot ?room1)))
)
```

Allows **exists** clause in precondition and goal

Typically not used in planners.
Better to add additional parameters

:universal-preconditions

```
(:action open-room
  :parameters
     (?robot ?room ~~?door1 ?door2~~)
  :precondition (and
     (not (in ?robot ?room))
     (forall (?d - DOOR)
          (closed ?d)
     ))
  :effect (and
     ;; something to do
     )
)
```

Allows **forall** clause in precondition and goal

:quantified-preconditions

Equivalent to adding both existential and universal preconditions (and goals)

```
(:requirements :quantified-preconditions)
```

=

```
(:requirements :existential-preconditions
               :universal-preconditions)
```

:conditional-effects

```
(:action lock-room
  :parameters
     (?robot ?room ?door1 ?door2)
  :precondition (and
      ;; check something
     )
  :effect (when
    ; Antecedent
    (and
          (is locked ?door1)
          (is locked ?door2)
    )
    ; Consequence
    (and (need-key))
)
```

:adl

- Stands for “Action Description Language”
- Equivalent to adding a lot of features

```
(:requirements :adl)
```

=

```
(:requirements
      :strips
      :typing
      :equality
      :disjunctive-preconditions
      :quantified-preconditions
      :conditional-effects
)
```