Planning Problem Representation

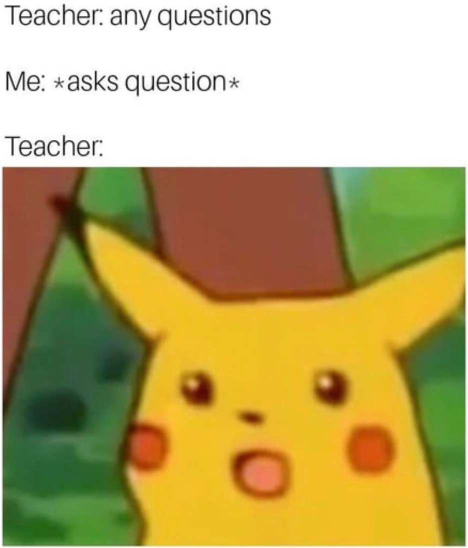
Problem representations + Assignment #1-2

Michaela Urbanovská

PUI Tutorial
Week 3

Lecture check

- Any questions regarding the lecture?



Feedback check

Thank you for your feedback!

- 5 responses
- Suggestions
  - Slow down the tutorials a bit
  - Everyone keeps up with the lecture with no problems

Problem Definitions

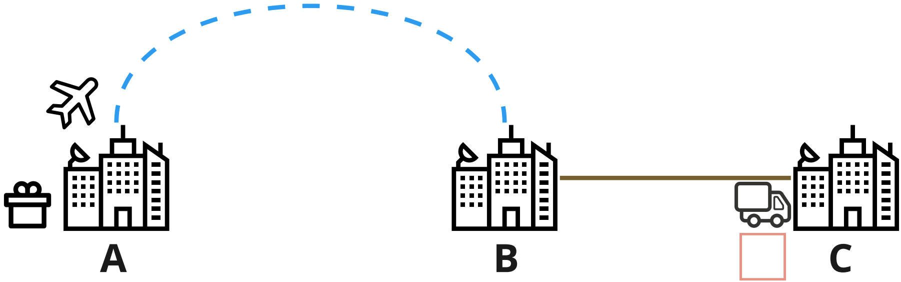
- **STRIPS**
- **FDR**
- **Specify** the model
- Representations used in planners with the search algorithms
- PDDL $\rightarrow$ Grounding $\rightarrow$ STRIPS/FDR

Grounding

- Process that creates **grounded** problem representation ready to be transformed into STRIPS, FDR, ...
- Many works on effective grounding, partial grounding, ...
- Can speeds up a planner significantly

Grounding

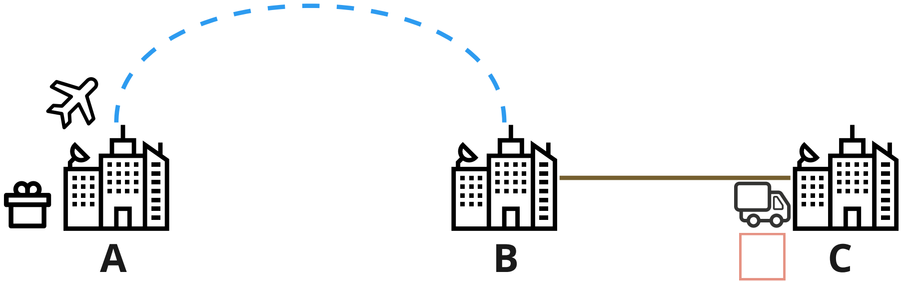
Let's create grounding for the example from the last time.



Grounding

(:types
    package vehicle - object
    location
    airplane truck - vehicle
)

(:objects
    A B C - location
    t - truck
    a - airplane
    p - package
)



Grounding

Ground all predicates

- Naive grounding → create all instances of predicates with existing objects

```
(:predicates
    (at ?o - object ?l - location)
    (in ?p - package ?v - vehicle)
    (road ?l1 - location ?l2 - location)
    (corridor ?l1 - location ?l2 - location)
    (empty ?v - vehicle)
)
```

Grounding

Full naive grounding of predicates

(at a A)
(at a B)
(at a C)
(at t A)
(at t B)
(at t C)
(at p A)
(at p B)
(at p C)
(empty a)
(empty t)
(in p a)
(in p t)

(road A B)
(road B A)
(road A A)
(road B B)
(road A C)
(road C A)
(road A A)
(road C C)
(road B C)
(road C B)
(road B B)
(road C C)

(corridor A B)
(corridor B A)
(corridor A A)
(corridor B B)
(corridor A C)
(corridor C A)
(corridor A A)
(corridor C C)
(corridor B C)
(corridor C B)
(corridor B B)
(corridor C C)

Grounding

Ground all actions

- Naive grounding → create all instances of actions with existing objects

(load ?p - package ?l - location ?v - vehicle)
(unload ?p - package ?l - location ?v - vehicle)
(drive ?t - truck ?l1 - location ?l2 - location)
(fly ?a - airplane ?l1 - location ?l2 - location)

Grounding

Full naive grounding of actions *(all preconditions and effects have to be grounded as well)*

(load p A t)
(load p B t)
(load p C t)
(load p A a)
(load p B a)
(load p C a)

(unload p A t)
(unload p B t)
(unload p C t)
(unload p A a)
(unload p B a)
(unload p C a)

(drive t A A)
(drive t A B)
(drive t A B)
(drive t B A)
(drive t B B)
(drive t B C)
(drive t C A)
(drive t C B)
(drive t C C)

(fly a A A)
(fly a A B)
(fly a A B)
(fly a B A)
(fly a B B)
(fly a B C)
(fly a C A)
(fly a C B)
(fly a C C)

Grounding

Full naive grounding of actions *(all preconditions and effects have to be grounded as well)*

		(drive t A A)	(fly a A A)
(load p A t)	(unload p A t)	(drive t A B)	(fly a A B)
(load p B t)	(unload p B t)	(drive t A B)	(fly a A B)
(load p C t)	(unload p C t)	(drive t B A)	(fly a B A)
(load p A a)	(unload p A a)	(drive t B B)	(fly a B B)
(load p B a)	(unload p B a)	(drive t B C)	(fly a B C)
(load p C a)	(unload p C a)	(drive t C A)	(fly a C A)
		(drive t C B)	(fly a C B)
		(drive t C C)	(fly a C C)

Now we have **full naive grounding** so we can start creating problem representations for planners!

STRIPS

STRIPS problem $\Pi = \langle F, O, s_{init}, s_{goal}, c \rangle$

- $F = \{f_1, f_2, \ldots f_n\}$ *(facts)*
- $O = \{o_1, o_2, \ldots o_m\}$ *(operators)*
- $s_{init} \subseteq F$ *(initial state)*
- $s_{goal} \subseteq F$ *(goal state specification)*
- $c(o_i) \in \mathbb{R}^+$ *(cost function)*

STRIPS

STRIPS problem $\Pi = \langle F, O, s_{init}, s_{goal}, c \rangle$

- $F = \{f_1, f_2, \dots f_n\}$ *(facts)*
- $O = \{o_1, o_2, \dots o_m\}$ *(operators)*
- $s_{init} \subseteq F$ *(initial state)*
- $s_{goal} \subseteq F$ *(goal state specification)*
- $c(o_i) \in \mathbb{R}^+$ *(cost function)*

STRIPS operator $o = \langle pre(o), add(o), del(o) \rangle$

- $pre(o) \subseteq F$ *(set of preconditions)*
- $add(o) \subseteq F$ *(set of add effects)*
- $del(o) \subseteq F$ *(set of delete effects)*

STRIPS

STRIPS problem $\Pi = \langle F, O, s_{init}, s_{goal}, c \rangle$

- $F = \{f_1, f_2, \dots f_n\}$ *(facts)*
- $O = \{o_1, o_2, \dots o_m\}$ *(operators)*
- $s_{init} \subseteq F$ *(initial state)*
- $s_{goal} \subseteq F$ *(goal state specification)*
- $c(o_i) \in \mathbb{R}^+$ *(cost function)*

STRIPS operator $o = \langle pre(o), add(o), del(o) \rangle$

- $pre(o) \subseteq F$ *(set of preconditions)*
- $add(o) \subseteq F$ *(set of add effects)*
- $del(o) \subseteq F$ *(set of delete effects)*
- operators are **well-formed**
  - $add(o) \cap del(o) = \emptyset$
  - $pre(o) \cap add(o) = \emptyset$

STRIPS

Applicable operator

Operator o is applicable in state s if $pre(o) \subseteq s$.
Resulting state $res(o, s) = (s \setminus del(o)) \cup add(o)$.
State s is a **goal state** iff $s_{goal} \subseteq s$.

STRIPS

Applicable operator

Operator o is applicable in state s if $pre(o) \subseteq s$.
Resulting state $res(o, s) = (s \setminus del(o)) \cup add(o)$.
State s is a **goal state** iff $s_{goal} \subseteq s$.

Sequence of applicable operators

Sequence of operators $\pi = \langle o_1, o_2, \ldots o_n \rangle$ is applicable in state s_0 if there are states $s_1, s_2, \ldots s_n$ such that o_i is applicable in s_{i-1} and $s_i = res(o_i, s_{i-1})$ for $1 \leq i \leq n$.

- $res(\pi, s_0) = s_n$ *(result of the applied operator sequence* π*)*
- $c(\pi) = \sum_{o \in \pi} c(o)$ *(cost of applying the operator sequence* π*)*

STRIPS

Applicable operator

Operator o is applicable in state s if $pre(o) \subseteq s$.
Resulting state $res(o, s) = (s \setminus del(o)) \cup add(o)$.
State s is a **goal state** iff $s_{goal} \subseteq s$.

Sequence of applicable operators

Sequence of operators $\pi = \langle o_1, o_2, \dots o_n \rangle$ is applicable in state s_0 if there are states $s_1, s_2, \dots s_n$ such that o_i is applicable in s_{i-1} and $s_i = res(o_i, s_{i-1})$ for $1 \leq i \leq n$.

- $res(\pi, s_0) = s_n$ *(result of the applied operator sequence π)*
- $c(\pi) = \sum_{o \in \pi} c(o)$ *(cost of applying the operator sequence π)*

Sequence π is called a **plan** if $s_{goal} \subseteq res(\pi, s_{init})$.

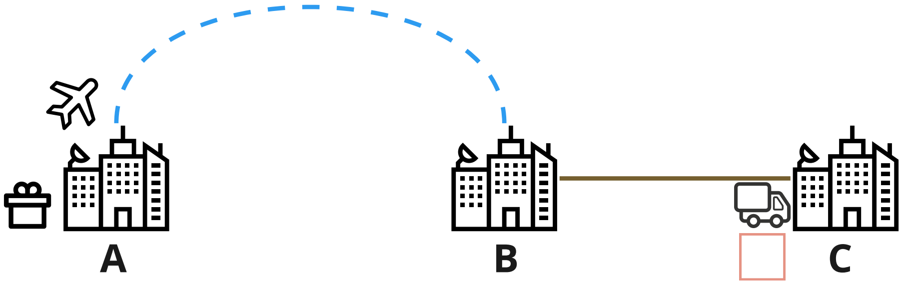
- π is an **optimal plan** is $c(\pi)$ is the minimal cost over all plans

Reachable state

State s is **reachable** if there exists an applicable sequence of operators π such that $res(\pi, s_{init} = s)$.
Set of all reachable states is denoted $\mathcal{R}_{\Pi}$.

STRIPS Example

Let's formulate STRIPS representation for the logistics problem.



$$\Pi = \langle F, O, s_{init}, s_{goal}, c \rangle$$

Grounding to STRIPS

Full naive grounding of predicates corresponds to STRIPS facts

(at a A) → a-A
(at a B) → a-B
(at a C) → a-C
(at t A) → t-A
(at t B) → t-B
(at t C) → t-C
(at p A) → p-A
(at p B) → p-B
(at p C) → p-C
(empty a) → emp-a
(empty t) → emp-t
(in p a) → p-a
(in p t) → p-t

(road A B) → r-A-B
(road B A) ...
(road A A) ...
(road B B) → r-B-B
(road A C) ...
(road C A) ...
(road A A) ...
(road C C) ...
(road B C) → r-B-C
(road C B) ...
(road B B) ...
(road C C) ...

(corridor A B) → c-A-B
(corridor B A) ...
(corridor A A) ...
(corridor B B) ...
(corridor A C) ...
(corridor C A) → c-C-A
(corridor A A) ...
(corridor C C) ...
(corridor B C) ...
(corridor C B) ...
(corridor B B) → c-B-B
(corridor C C) ...

Grounding to STRIPS

Full naive grounding of actions can be transformed into STRIPS operators

		(drive t A A)	(fly a A A)
(load p A t)	(unload p A t)	(drive t A B)	(fly a A B)
(load p B t)	(unload p B t)	(drive t A B)	(fly a A B)
(load p C t)	(unload p C t)	(drive t B A)	(fly a B A)
(load p A a)	(unload p A a)	(drive t B B)	(fly a B B)
(load p B a)	(unload p B a)	(drive t B C)	(fly a B C)
(load p C a)	(unload p C a)	(drive t C A)	(fly a C A)
		(drive t C B)	(fly a C B)
		(drive t C C)	(fly a C C)

STRIPS Example

Lifted action

```
(:action load
    :parameters (
        ?p - package
        ?l - location
        ?v - vehicle)
    :precondition (and
        (at ?p ?l)
        (at ?v ?l)
        (empty ?v)
    )
    :effect (and
        (not (at ?p ?l))
        (in ?p ?v)
        (not (empty ?v))
    )
)
```

STRIPS Example

Lifted action

```
(:action load
    :parameters (
        ?p - package
        ?l - location
        ?v - vehicle)
    :precondition (and
        (at ?p ?l)
        (at ?v ?l)
        (empty ?v)
    )
    :effect (and
        (not (at ?p ?l))
        (in ?p ?v)
        (not (empty ?v))
    )
)
```

Grounded action

```
(:action load
    :parameters (
        p - package
        A - location
        t - vehicle)
    :precondition (and
        (at p A)
        (at t A)
        (empty t)
    )
    :effect (and
        (not (at p A))
        (in p t)
        (not (empty t))
    )
)
```

STRIPS Example

Grounded action

```
(:action load
    :parameters (
        p - package
        A - location
        t - vehicle)
    :precondition (and
        (at p A)
        (at t A)
        (empty t)
    )
    :effect (and
        (not (at p A))
        (in p t)
        (not (empty t))
    )
)
```

STRIPS Example

Grounded action

```
(:action load
    :parameters (
        p - package
        A - location
        t - vehicle)
    :precondition (and
        (at p A)
        (at t A)
        (empty t)
    )
    :effect (and
        (not (at p A))
        (in p t)
        (not (empty t))
    )
)
```

STRIPS operator load-p-A-t

pre(load-p-A-t) $= \{\}$
add(load-p-A-t) $= \{\}$
del(load-p-A-t) $= \{\}$

STRIPS Example

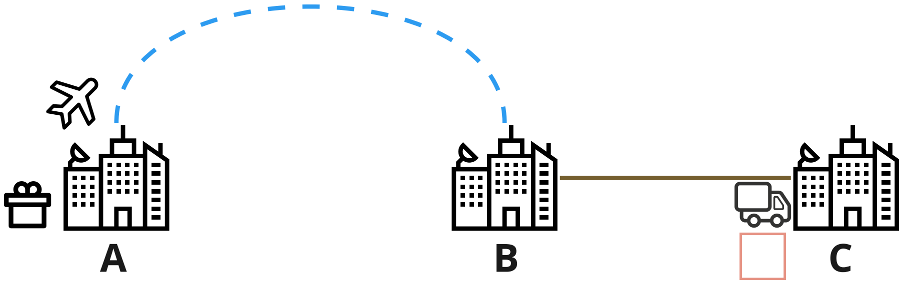
Grounded action

```
(:action load
    :parameters (
        p - package
        A - location
        t - vehicle)
    :precondition (and
        (at p A)
        (at t A)
        (empty t)
    )
    :effect (and
        (not (at p A))
        (in p t)
        (not (empty t))
    )
)
```

STRIPS operator load-p-A-t

pre(load-p-A-t) $=$ {p-A, t-A, emp-t}
add(load-p-A-t) $=$ {p-t}
del(load-p-A-t) $=$ {p-A, emp-t}

STRIPS Example



$$\Pi = \langle F, O, s_{init}, s_{goal}, c \rangle$$

F = {a-A, a-B, ..., t-A, ..., p-A, ..., emp-a, emp-t, r-A-A, ..., c-A-A, ...}
O = {load-p-A-t, ..., unload-p-A-t, ..., drive-t-A-A, ..., fly-a-A-A, ...}
s_{init} = {p-A, a-A, t-C, c-A-B, c-B-A, r-B-C, r-C-B}
s_{goal} = {p-C}

Problem definitions

FDR

FDR problem $P = \langle \mathcal{V}, \mathcal{O}, s_{init}, s_{goal}, c \rangle$

- $\mathcal{V} = \{V_1, V_2, \ldots V_n\}$ *(finite set of variables)*
- $\mathcal{O} = \{o_1, o_2, \ldots o_m\}$ *(set of operators)*
- s_{init} *(initial state)*
- s_{goal} *(goal state)*
- $c(o_i) \in \mathbb{R}^+$

FDR

FDR problem $P = \langle \mathcal{V}, \mathcal{O}, s_{init}, s_{goal}, c \rangle$

- $\mathcal{V}$ *(finite set of variables)*
  - $V \in \mathcal{V}$ *(variable)*
  - D_V *(finite domain of variable V)*
- s *(state)* is partial variable assignment over $\mathcal{V}$
  - vars(s) $= V \in \mathcal{V}$ assigned in s
  - $s[V] =$ value of V in s
  - s is **consistent** with s' if $s[V] = s'[V]$ for all $V \in vars(s')$
  - atom $V = v$ is true in s if $s[V] = v$

FDR

FDR operator $o = \langle pre(o), eff(o) \rangle$

- $\mathcal{O}$ *(set of operators)*
  - $pre(o) =$ partial assignment over $\mathcal{V}$ *(preconditions)*
  - $eff(o) =$ partial assignment over $\mathcal{V}$ *(effects)*
  - $V = v$ cannot be in both pre(o) and eff(o)

FDR

Applicable operator

Operator o is applicable in state s if $pre(o)$ is **consistent** with s.

Resulting state $res(o, s) = \begin{cases} eff(o)[V], & \text{if } V \in vars(eff(o)) \\ s[V], & \text{otherwise} \end{cases}$

FDR

Applicable operator

Operator o is applicable in state s if $pre(o)$ is **consistent** with s.

Resulting state $res(o, s) = \begin{cases} eff(o)[V], & \text{if } V \in vars(eff(o)) \\ s[V], & \text{otherwise} \end{cases}$

Sequence of applicable operators

Sequence of operators $\pi = \langle o_1, o_2, \ldots o_n \rangle$ is applicable in state s_0 if there are state $s_1, s_2, \ldots s_n$ such that o_i is applicable in s_{i-1} and $s_i = res(o_i, s_{i-1})$ for $1 \leq i \leq n$.

- $res(\pi, s_0) = s_n$ *(result of the applied operator sequence π)*
- $c(\pi) = \sum_{o \in \pi} c(o)$ *(cost of applying the operator sequence π)*

FDR

Applicable operator

Operator o is applicable in state s if $pre(o)$ is **consistent** with s.

Resulting state $res(o, s) = \begin{cases} eff(o)[V], & \text{if } V \in vars(eff(o))) \\ s[V], & \text{otherwise} \end{cases}$

Sequence of applicable operators

Sequence of operators $\pi = \langle o_1, o_2, \dots o_n \rangle$ is applicable in state s_0 if there are state $s_1, s_2, \dots s_n$ such that o_i is applicable in s_{i-1} and $s_i = res(o_i, s_{i-1})$ for $1 \leq i \leq n$.

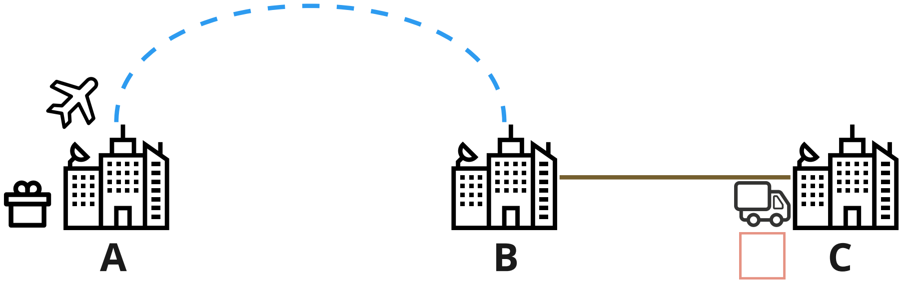
- $res(\pi, s_0) = s_n$ *(result of the applied operator sequence π)*
- $c(\pi) = \sum_{o \in \pi} c(o)$ *(cost of applying the operator sequence π)*

Sequence π is called a **plan** if $res(\pi, s_{init})$ is consistent with s_{goal}.

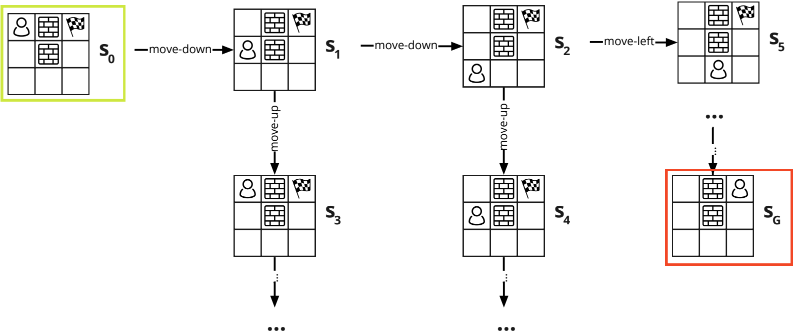
- π is an **optimal plan** is $c(\pi)$ is the minimal cost over all plans

FDR Example

Let's model the logistics example using FDR.



- Both STRIP and FDR have a notion of **state** and **operator**
  - s_0 is the initial state which gets expanded by using $o \in O$ creating new state $s' \rightarrow$ **transition system**



Assignment #1-2 - Grounding

- Second part of the Assignment #1
- **Task:** implement a grounder for parsed PDDL files that will be base for the STRIPS representation in your planner
- **Points:** maximum 10
- **Deadlines**
  - 20.3.2023 - 23:59 (Monday)
  - 22.3.2023 - 23:59 (Wednesday)

All information is available on Courseware

Recap

- You know how to create naive grounding
- You know how to construct STRIPS and FDR representations
- You should be able to implement Assignment 1-2 - Grounding

The End

Feedback form