

# Representations

Michal Štolba

stolba@agents.fel.cvut.cz



## PUI (Planning in Artificial Intelligence)

# STRIPS

(Stanford Research Institute Problem Solver)

- ▶ 1966-1972 – Shakey the Robot

- ▶  $\langle P, O, I, G \rangle$

$P$ : finite set of propositional (true/false) variables

$O$ : finite set of operators

initial state ( $p \in P \wedge p = \text{true}$ , other false)

goal state ( $p \in P \wedge p = \text{true}; p' \in P \wedge p' = \text{false}$ )

- ▶ Set representation

(true/false determined by the set membership)

- ▶ Plan existence PSPACE-Complete

# STRIPS

(Stanford Research Institute Problem Solver)

- ▶ 1966-1972 – Shakey the Robot

- ▶  $\langle P, O, I, G \rangle$

$P$  finite set of propositional (true/false) variables

$O$  finite set of operators:

$\langle p \in P \mid p = \text{true} \rangle$

$\langle p \in P \mid p = \text{false} \rangle$

$\langle p \in P \mid p = \text{true} \rangle$

$I$  initial state ( $p \in P$  s.t.  $p = \text{true}$ , other false)

$G$  goal state ( $p \in P$  s.t.  $p = \text{true}$ ;  $p' \in P$  s.t.  $p = \text{false}$ )

- ▶ Set representation

(true/false determined by the set membership)

- ▶ Plan existence PSPACE-Complete

# STRIPS

(Stanford Research Institute Problem Solver)

- ▶ 1966-1972 – Shakey the Robot

- ▶  $\langle P, O, I, G \rangle$

$P$  finite set of propositional (true/false) variables

$O$  finite set of operators:

$\{p \in P \mid p = \text{true}\}$

$\{p \in P \mid p = \text{true}\}$

$\{p \in P \mid p = \text{true}\}$

$I$  initial state ( $p \in P$  s.t.  $p = \text{true}$ , other false)

$G$  goal state ( $p \in P$  s.t.  $p = \text{true}$ ;  $p' \in P$  s.t.  $p = \text{false}$ )

- ▶ Set representation

(true/false represented by the set membership)

- ▶ Plan existence PSPACE-Complete

# STRIPS

(Stanford Research Institute Problem Solver)

- ▶ 1966-1972 – Shakey the Robot

- ▶  $\langle P, O, I, G \rangle$

$P$  finite set of propositional (true/false) variables

$O$  finite set of operators:

$\langle p \in P, p = \text{true} \rangle$

$\langle p \in P, p = \text{true} \rangle$

$\langle p \in P, p = \text{false} \rangle$

$I$  initial state ( $p \in P$  s.t.  $p = \text{true}$ , other false)

$G$  goal state ( $p \in P$  s.t.  $p = \text{true}$ ;  $p' \in P$  s.t.  $p = \text{false}$ )

- ▶ Set representation

(true/false represented by the set membership)

- ▶ Plan existence PSPACE-Complete

# STRIPS

(Stanford Research Institute Problem Solver)

- ▶ 1966-1972 – Shakey the Robot

- ▶  $\langle P, O, I, G \rangle$

$P$  finite set of propositional (true/false) variables

$O$  finite set of operators:

pre:  $\{p \in P \mid p = \text{true}\}$

add:  $\{p \in P \mid p \leftarrow \text{true}\}$

del:  $\{p \in P \mid p \leftarrow \text{false}\}$

$I$  initial state ( $p \in P$  s.t.  $p = \text{true}$ , other false)

$G$  goal state ( $p \in P$  s.t.  $p = \text{true}$ ;  $p' \in P$  s.t.  $p = \text{false}$ )

- ▶ Set representation

$\{p \in P \mid p = \text{true}\} \cup \{p' \in P \mid p' = \text{false}\}$

- ▶ Plan existence PSPACE-Complete

# STRIPS

(Stanford Research Institute Problem Solver)

- ▶ 1966-1972 – Shakey the Robot

- ▶  $\langle P, O, I, G \rangle$

$P$  finite set of propositional (true/false) variables

$O$  finite set of operators:

pre:  $\{p \in P \mid p = \text{true}\}$

add:  $\{p \in P \mid p \leftarrow \text{true}\}$

del:  $\{p \in P \mid p \leftarrow \text{false}\}$

$I$  initial state ( $p \in P$  s.t.  $p = \text{true}$ , other false)

$G$  goal state ( $p \in P$  s.t.  $p = \text{true}$ ;  $p' \in P$  s.t.  $p = \text{false}$ )

- ▶ Set representation

$\{p \in P \mid p = \text{true}\} \subseteq P$

- ▶ Plan existence PSPACE-Complete

# STRIPS

(Stanford Research Institute Problem Solver)

▶ 1966-1972 – Shakey the Robot

▶  $\langle P, O, I, G \rangle$

$P$  finite set of propositional (true/false) variables

$O$  finite set of operators:

pre:  $\{p \in P \mid p = \text{true}\}$

add:  $\{p \in P \mid p \leftarrow \text{true}\}$

del:  $\{p \in P \mid p \leftarrow \text{false}\}$

$I$  initial state ( $p \in P$  s.t.  $p = \text{true}$ , other false)

$G$  goal state ( $p \in P$  s.t.  $p = \text{true}$ ;  $p' \in P$  s.t.  $p = \text{false}$ )

▶ Set representation

▶ Plan existence PSPACE-Complete



# STRIPS

(Stanford Research Institute Problem Solver)

▶ 1966-1972 – Shakey the Robot

▶  $\langle P, O, I, G \rangle$

$P$  finite set of propositional (true/false) variables

$O$  finite set of operators:

**pre:**  $\{p \in P \mid p = \text{true}\}$

**add:**  $\{p \in P \mid p \leftarrow \text{true}\}$

**del:**  $\{p \in P \mid p \leftarrow \text{false}\}$

$I$  initial state ( $p \in P$  s.t.  $p = \text{true}$ , other false)

$G$  goal state ( $p \in P$  s.t.  $p = \text{true}$ ;  $p' \in P$  s.t.  $p = \text{false}$ )

▶ Set representation

▶ Plan existence PSPACE-Complete

# STRIPS

(Stanford Research Institute Problem Solver)

▶ 1966-1972 – Shakey the Robot

▶  $\langle P, O, I, G \rangle$

$P$  finite set of propositional (true/false) variables

$O$  finite set of operators:

**pre:**  $\{p \in P \mid p = \text{true}\}$

**add:**  $\{p \in P \mid p \leftarrow \text{true}\}$

**del:**  $\{p \in P \mid p \leftarrow \text{false}\}$

$I$  initial state ( $p \in P$  s.t.  $p = \text{true}$ , other false)

$G$  goal state ( $p \in P$  s.t.  $p = \text{true}$ ;  $p' \in P$  s.t.  $p = \text{false}$ )

▶ Set representation

▶ Plan existence PSPACE-Complete

# STRIPS

(Stanford Research Institute Problem Solver)

▶ 1966-1972 – Shakey the Robot

▶  $\langle P, O, I, G \rangle$

$P$  finite set of propositional (true/false) variables

$O$  finite set of operators:

**pre:**  $\{p \in P \mid p = \text{true}\}$

**add:**  $\{p \in P \mid p \leftarrow \text{true}\}$

**del:**  $\{p \in P \mid p \leftarrow \text{false}\}$

$I$  initial state ( $p \in P$  s.t.  $p = \text{true}$ , other false)

$G$  goal state ( $p \in P$  s.t.  $p = \text{true}$ ;  $p' \in P$  s.t.  $p = \text{false}$ )

▶ Set representation

▶ Plan existence PSPACE-Complete

# STRIPS

(Stanford Research Institute Problem Solver)

▶ 1966-1972 – Shakey the Robot

▶  $\langle P, O, I, G \rangle$

$P$  finite set of propositional (true/false) variables

$O$  finite set of operators:

pre:  $\{p \in P \mid p = \text{true}\}$

add:  $\{p \in P \mid p \leftarrow \text{true}\}$

del:  $\{p \in P \mid p \leftarrow \text{false}\}$

$I$  initial state ( $p \in P$  s.t.  $p = \text{true}$ , other false)

$G$  goal state ( $p \in P$  s.t.  $p = \text{true}$ ;  $p' \in P$  s.t.  $p = \text{false}$ )

▶ Set representation

▶ Plan existence PSPACE-Complete

# STRIPS

(Stanford Research Institute Problem Solver)

- ▶ 1966-1972 – Shakey the Robot
- ▶  $\langle P, O, I, G \rangle$ 
  - $P$  finite set of propositional (true/false) variables
  - $O$  finite set of operators:
    - pre:  $\{p \in P \mid p = \text{true}\}$
    - add:  $\{p \in P \mid p \leftarrow \text{true}\}$
    - del:  $\{p \in P \mid p \leftarrow \text{false}\}$
  - $I$  initial state ( $p \in P$  s.t.  $p = \text{true}$ , other false)
  - $G$  goal state ( $p \in P$  s.t.  $p = \text{true}$ ;  $p' \in P$  s.t.  $p = \text{false}$ )

▶ Set representation

▶ Plan existence PSPACE-Complete

# STRIPS

(Stanford Research Institute Problem Solver)

- ▶ 1966-1972 – Shakey the Robot
- ▶  $\langle P, O, I, G \rangle$ 
  - $P$  finite set of propositional (true/false) variables
  - $O$  finite set of operators:
    - pre:  $\{p \in P \mid p = \text{true}\}$
    - add:  $\{p \in P \mid p \leftarrow \text{true}\}$
    - del:  $\{p \in P \mid p \leftarrow \text{false}\}$
  - $I$  initial state ( $p \in P$  s.t.  $p = \text{true}$ , other false)
  - $G$  goal state ( $p \in P$  s.t.  $p = \text{true}$ ;  $p' \in P$  s.t.  $p = \text{false}$ )

▶ Set representation

▶ Plan existence PSPACE-Complete

# STRIPS

(Stanford Research Institute Problem Solver)

- ▶ 1966-1972 – Shakey the Robot
- ▶  $\langle P, O, I, G \rangle$ 
  - $P$  finite set of propositional (true/false) variables
  - $O$  finite set of operators:
    - pre:  $\{p \in P \mid p = \text{true}\}$
    - add:  $\{p \in P \mid p \leftarrow \text{true}\}$
    - del:  $\{p \in P \mid p \leftarrow \text{false}\}$
  - $I$  initial state ( $p \in P$  s.t.  $p = \text{true}$ , other false)
  - $G$  goal state ( $p \in P$  s.t.  $p = \text{true}$ ;  $p' \in P$  s.t.  $p = \text{false}$ )
- ▶ Set representation
  - ▶ true/false determined by the set membership
- ▶ Plan existence PSPACE-Complete

# STRIPS

(Stanford Research Institute Problem Solver)

- ▶ 1966-1972 – Shakey the Robot
- ▶  $\langle P, O, I, G \rangle$ 
  - $P$  finite set of propositional (true/false) variables
  - $O$  finite set of operators:
    - pre:  $\{p \in P \mid p = \text{true}\}$
    - add:  $\{p \in P \mid p \leftarrow \text{true}\}$
    - del:  $\{p \in P \mid p \leftarrow \text{false}\}$
  - $I$  initial state ( $p \in P$  s.t.  $p = \text{true}$ , other false)
  - $G$  goal state ( $p \in P$  s.t.  $p = \text{true}$ ;  $p' \in P$  s.t.  $p = \text{false}$ )
- ▶ Set representation
  - ▶ true/false determined by the set membership

▶ Plan existence PSPACE-Complete



# STRIPS

(Stanford Research Institute Problem Solver)

- ▶ 1966-1972 – Shakey the Robot
- ▶  $\langle P, O, I, G \rangle$ 
  - $P$  finite set of propositional (true/false) variables
  - $O$  finite set of operators:
    - pre:  $\{p \in P \mid p = \text{true}\}$
    - add:  $\{p \in P \mid p \leftarrow \text{true}\}$
    - del:  $\{p \in P \mid p \leftarrow \text{false}\}$
  - $I$  initial state ( $p \in P$  s.t.  $p = \text{true}$ , other false)
  - $G$  goal state ( $p \in P$  s.t.  $p = \text{true}$ ;  $p' \in P$  s.t.  $p = \text{false}$ )
- ▶ Set representation
  - ▶ true/false determined by the set membership

▶ Plan existence PSPACE-Complete

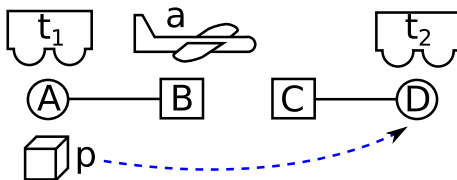
# STRIPS

(Stanford Research Institute Problem Solver)

- ▶ 1966-1972 – Shakey the Robot
- ▶  $\langle P, O, I, G \rangle$ 
  - $P$  finite set of propositional (true/false) variables
  - $O$  finite set of operators:
    - pre:  $\{p \in P \mid p = \text{true}\}$
    - add:  $\{p \in P \mid p \leftarrow \text{true}\}$
    - del:  $\{p \in P \mid p \leftarrow \text{false}\}$
  - $I$  initial state ( $p \in P$  s.t.  $p = \text{true}$ , other false)
  - $G$  goal state ( $p \in P$  s.t.  $p = \text{true}$ ;  $p' \in P$  s.t.  $p = \text{false}$ )
- ▶ Set representation
  - ▶ true/false determined by the set membership
- ▶ Plan existence PSPACE-Complete

# STRIPS

## Example



# STRIPS

## Example

- ▶  $P$  propositions:
  - ▶ truck-at-A, truck-at-B,
  - ▶ plane-at-B, plane-at-C,
  - ▶ package-at-A, package-at-B, package-at-C,
  - ▶ package-in-t, package-in-a
- ▶  $2^9 = 512$  states

# STRIPS

## Example

- ▶  $P$  propositions:
  - ▶ truck-at-A, truck-at-B,
  - ▶ plane-at-B, plane-at-C,
  - ▶ package-at-A, package-at-B, package-at-C,
  - ▶ package-in-t, package-in-a
- ▶  $2^9 = 512$  states

# STRIPS

## Example

- ▶  $P$  propositions:
  - ▶ truck-at-A, truck-at-B,
  - ▶ plane-at-B, plane-at-C,
  - ▶ package-at-A, package-at-B, package-at-C,
  - ▶ package-in-t, package-in-a
- ▶  $2^9 = 512$  states

# STRIPS

## Example

- ▶  $P$  propositions:
  - ▶ truck-at-A, truck-at-B,
  - ▶ plane-at-B, plane-at-C,
  - ▶ package-at-A, package-at-B, package-at-C,
  - ▶ package-in-t, package-in-a
- ▶  $2^9 = 512$  states

# STRIPS

## Example

- ▶  $P$  propositions:
  - ▶ truck-at-A, truck-at-B,
  - ▶ plane-at-B, plane-at-C,
  - ▶ package-at-A, package-at-B, package-at-C,
  - ▶ package-in-t, package-in-a
- ▶  $2^9 = 512$  states



# STRIPS

## Example

- ▶  $P$  propositions:
  - ▶ truck-at-A, truck-at-B,
  - ▶ plane-at-B, plane-at-C,
  - ▶ package-at-A, package-at-B, package-at-C,
  - ▶ package-in-t, package-in-a
- ▶  $2^9 = 512$  states

# STRIPS

## Example

▶ *O* operators:

▶ load-p-a-B

```
pre: {plane-at-B, package-at-B}
add: {package-in-a}
del: {package-at-B}
```

▶ move-t-A-B

```
pre: {truck-at-A}
add: {truck-at-B}
del: {truck-at-A}
```

# STRIPS

## Example

▶ *O* operators:

▶ load-p-a-B

pre: {plane-at-B, package-at-B}

add: {package-in-a}

del: {package-at-B}

▶ move-t-A-B

pre: {truck-at-A}

add: {truck-at-B}

del: {truck-at-A}

# STRIPS

## Example

▶ 0 operators:

▶ load-p-a-B

pre: {plane-at-B, package-at-B}

add: {package-in-a}

del: {package-at-B}

▶ move-t-A-B

pre: {truck-at-A}

add: {truck-at-B}

del: {truck-at-A}

# STRIPS

## Example

▶ 0 operators:

▶ load-p-a-B

pre: {plane-at-B, package-at-B}

add: {package-in-a}

del: {package-at-B}

▶ move-t-A-B

pre: {truck-at-A}

add: {truck-at-B}

del: {truck-at-A}

# STRIPS

## Example

▶ 0 operators:

▶ load-p-a-B

pre: {plane-at-B, package-at-B}

add: {package-in-a}

del: {package-at-B}

▶ move-t-A-B

pre: {truck-at-A}

add: {truck-at-B}

del: {truck-at-A}

# STRIPS

## Example

▶ *O* operators:

▶ load-p-a-B

pre: {plane-at-B, package-at-B}

add: {package-in-a}

del: {package-at-B}

▶ move-t-A-B

pre: {truck-at-A}

add: {truck-at-B}

del: {truck-at-A}

# STRIPS

## Example

▶ *O* operators:

▶ load-p-a-B

pre: {plane-at-B, package-at-B}

add: {package-in-a}

del: {package-at-B}

▶ move-t-A-B

pre: {truck-at-A}

add: {truck-at-B}

del: {truck-at-A}



# STRIPS

## Example

▶  $O$  operators:

▶ load-p-a-B

pre: {plane-at-B, package-at-B}

add: {package-in-a}

del: {package-at-B}

▶ move-t-A-B

pre: {truck-at-A}

add: {truck-at-B}

del: {truck-at-A}

# STRIPS

## Example

▶  $O$  operators:

▶ load-p-a-B

pre: {plane-at-B, package-at-B}

add: {package-in-a}

del: {package-at-B}

▶ move-t-A-B

pre: {truck-at-A}

add: {truck-at-B}

del: {truck-at-A}

# MPT/FDR

## Multi-valued Planning Task / Finite Domain Representation (or SAS+)

▶  $\langle V, O, i, g \rangle$

$V$  finite set of state variables  $v$  with associated finite domain  $D_v$

- ▶ If  $D_v = \{\text{true}, \text{false}\}$  equivalent to STRIPS
- ▶ Partial state  $s$  defined on  $V' \subseteq V$  variables is a function  $s : v_1 \times \dots \times v_k \mapsto D_{v_1} \times \dots \times D_{v_k}$
- ▶ State is a partial state defined on all variables in  $V$

$i$  The initial state

$g$  Partial state defining the goal

# MPT/FDR

## Multi-valued Planning Task / Finite Domain Representation (or SAS+)

▶  $\langle V, O, i, g \rangle$

$V$  finite set of state variables  $v$  with associated finite domain  $D_v$

- ▶ If  $D_v = \{\text{true}, \text{false}\}$  equivalent to STRIPS
- ▶ Partial state  $s$  defined on  $V' \subseteq V$  variables is a function  $s : v_1 \times \dots \times v_k \mapsto D_{v_1} \times \dots \times D_{v_k}$
- ▶ State is a partial state defined on all variables in  $V$

$i$  The initial state

$g$  Partial state defining the goal

# MPT/FDR

## Multi-valued Planning Task / Finite Domain Representation (or SAS+)

▶  $\langle V, O, i, g \rangle$

$V$  finite set of state variables  $v$  with associated finite domain  $D_v$

- ▶ If  $D_v = \{\text{true}, \text{false}\}$  equivalent to STRIPS
- ▶ Partial state  $s$  defined on  $V' \subseteq V$  variables is a function  $s : v_1 \times \dots \times v_k \mapsto D_{v_1} \times \dots \times D_{v_k}$
- ▶ State is a partial state defined on all variables in  $V$

$i$  The initial state

$g$  Partial state defining the goal

# MPT/FDR

## Multi-valued Planning Task / Finite Domain Representation (or SAS+)

▶  $\langle V, O, i, g \rangle$

$V$  finite set of state variables  $v$  with associated finite domain  $D_v$

- ▶ If  $D_v = \{\text{true}, \text{false}\}$  equivalent to STRIPS
- ▶ Partial state  $s$  defined on  $V' \subseteq V$  variables is a function  $s : v_1 \times \dots \times v_k \mapsto D_{v_1} \times \dots \times D_{v_k}$
- ▶ State is a partial state defined on all variables in  $V$

$i$  The initial state

$g$  Partial state defining the goal

# MPT/FDR

## Multi-valued Planning Task / Finite Domain Representation (or SAS+)

▶  $\langle V, O, i, g \rangle$

$V$  finite set of state variables  $v$  with associated finite domain  $D_v$

- ▶ If  $D_v = \{\text{true}, \text{false}\}$  equivalent to STRIPS
- ▶ Partial state  $s$  defined on  $V' \subseteq V$  variables is a function  $s : v_1 \times \dots \times v_k \mapsto D_{v_1} \times \dots \times D_{v_k}$
- ▶ State is a partial state defined on all variables in  $V$

$i$  The initial state

$g$  Partial state defining the goal

# MPT/FDR

## Multi-valued Planning Task / Finite Domain Representation (or SAS+)

▶  $\langle V, O, i, g \rangle$

$V$  finite set of state variables  $v$  with associated finite domain  $D_v$

- ▶ If  $D_v = \{\text{true}, \text{false}\}$  equivalent to STRIPS
- ▶ Partial state  $s$  defined on  $V' \subseteq V$  variables is a function  $s : v_1 \times \dots \times v_k \mapsto D_{v_1} \times \dots \times D_{v_k}$
- ▶ State is a partial state defined on all variables in  $V$

$i$  The initial state

$g$  Partial state defining the goal



# MPT/FDR

## Multi-valued Planning Task / Finite Domain Representation (or SAS+)

▶  $\langle V, O, i, g \rangle$

$V$  finite set of state variables  $v$  with associated finite domain  $D_v$

- ▶ If  $D_v = \{\text{true}, \text{false}\}$  equivalent to STRIPS
- ▶ Partial state  $s$  defined on  $V' \subseteq V$  variables is a function  $s : v_1 \times \dots \times v_k \mapsto D_{v_1} \times \dots \times D_{v_k}$
- ▶ State is a partial state defined on all variables in  $V$

$i$  The initial state

$g$  Partial state defining the goal

# MPT/FDR

## Multi-valued Planning Task / Finite Domain Representation (or SAS+)

▶  $\langle V, O, i, g \rangle$

$V$  finite set of state variables  $v$  with associated finite domain  $D_v$

- ▶ If  $D_v = \{\text{true}, \text{false}\}$  equivalent to STRIPS
- ▶ Partial state  $s$  defined on  $V' \subseteq V$  variables is a function  $s : v_1 \times \dots \times v_k \mapsto D_{v_1} \times \dots \times D_{v_k}$
- ▶ State is a partial state defined on all variables in  $V$

$i$  The initial state

$g$  Partial state defining the goal

# MPT/FDR

## Multi-valued Planning Task / Finite Domain Representation (or SAS+)

▶  $\langle V, O, i, g \rangle$

○ finite set of operators  $o = \langle \text{pre}(o), \text{eff}(o) \rangle$  where

pre: partial state

eff: partial state

▶ Application of  $o$  in state  $s$  resulting in  $s'$

Values of variables defined in  $\text{pre}(o)$  must be equal to their values in  $s$ .

Values of variables in  $o$  are

replaced by values in  $\text{eff}(o)$  if defined

and to values in  $s$  else.

# MPT/FDR

## Multi-valued Planning Task / Finite Domain Representation (or SAS+)

▶  $\langle V, O, i, g \rangle$

○ finite set of operators  $o = \langle \text{pre}(o), \text{eff}(o) \rangle$  where

pre: partial state

eff: partial state

▶ Application of  $o$  in state  $s$  resulting in  $s'$

Values of variables defined in  $\text{pre}(o)$  must be equal to their values in  $s$

Values of variables in  $o$  are

replaced by values in  $\text{eff}(o)$  if present

else to values in  $s$

# MPT/FDR

## Multi-valued Planning Task / Finite Domain Representation (or SAS+)

▶  $\langle V, O, i, g \rangle$

○ finite set of operators  $o = \langle \text{pre}(o), \text{eff}(o) \rangle$  where

**pre:** partial state

**eff:** partial state

▶ Application of  $o$  in state  $s$  resulting in  $s'$

Values of variables defined in  $\text{pre}(o)$  must be equal to their values in  $s$

Values of variables in  $o$  are

replaced by values in  $\text{eff}(o)$  if present

else they remain the same

# MPT/FDR

## Multi-valued Planning Task / Finite Domain Representation (or SAS+)

▶  $\langle V, O, i, g \rangle$

○ finite set of operators  $o = \langle \text{pre}(o), \text{eff}(o) \rangle$  where

**pre:** partial state

**eff:** partial state

▶ Application of  $o$  in state  $s$  resulting in  $s'$

Value of  $s'$  is  $s$  (undefined in  $s'$ ) unless  $o$  specifies otherwise

Value of  $s'$  is  $o$  unless  $o$  specifies otherwise

Value of  $s'$  is  $o$  unless  $o$  specifies otherwise

Value of  $s'$  is  $o$  unless  $o$  specifies otherwise

# MPT/FDR

## Multi-valued Planning Task / Finite Domain Representation (or SAS+)

▶  $\langle V, O, i, g \rangle$

○ finite set of operators  $o = \langle \text{pre}(o), \text{eff}(o) \rangle$  where

**pre:** partial state

**eff:** partial state

▶ Application of  $o$  in state  $s$  resulting in  $s'$

Value of  $s'$  is  $s$  (undefined in  $s'$ ) unless  $o$  specifies otherwise

Value of  $s'$  is  $o$  unless  $o$  specifies otherwise

Value of  $s'$  is  $o$  unless  $o$  specifies otherwise

Value of  $s'$  is  $o$  unless  $o$  specifies otherwise

# MPT/FDR

## Multi-valued Planning Task / Finite Domain Representation (or SAS+)

▶  $\langle V, O, i, g \rangle$

○ finite set of operators  $o = \langle \text{pre}(o), \text{eff}(o) \rangle$  where

**pre:** partial state

**eff:** partial state

▶ Application of  $o$  in state  $s$  resulting in  $s'$

- ▶ Values of variables defined in  $\text{pre}(o)$  must be equal to their values in  $s$
- ▶ Values in of variables in  $s'$  are set to values in  $\text{eff}(o)$  if defined set to values in  $s$  else



# MPT/FDR

## Multi-valued Planning Task / Finite Domain Representation (or SAS+)

▶  $\langle V, O, i, g \rangle$

○ finite set of operators  $o = \langle \text{pre}(o), \text{eff}(o) \rangle$  where

**pre:** partial state

**eff:** partial state

▶ Application of  $o$  in state  $s$  resulting in  $s'$

▶ Values of variables defined in  $\text{pre}(o)$  must be equal to their values in  $s$

▶ Values of variables in  $s'$  are  
set to values in  $\text{eff}(o)$  if defined  
set to values in  $s$  else

# MPT/FDR

## Multi-valued Planning Task / Finite Domain Representation (or SAS+)

▶  $\langle V, O, i, g \rangle$

○ finite set of operators  $o = \langle \text{pre}(o), \text{eff}(o) \rangle$  where

**pre:** partial state

**eff:** partial state

▶ Application of  $o$  in state  $s$  resulting in  $s'$

- ▶ Values of variables defined in  $\text{pre}(o)$  must be equal to their values in  $s$
- ▶ Values in of variables in  $s'$  are set to values in  $\text{eff}(o)$  if defined set to values in  $s$  else

# MPT/FDR

## Example

### V variables and their domains:

- ▶ truck-at  $\in \{A, B\}$
- ▶ plane-at  $\in \{B, C\}$
- ▶ package-at  $\in \{A, B, C, t, a\}$
- ▶  $2 \times 2 \times 5 = 20$  states

### O operators:

truck-at := A, plane-at := B, package-at := A  
truck-at := B, plane-at := C, package-at := B  
truck-at := A, plane-at := C, package-at := A  
truck-at := B, plane-at := A, package-at := B  
truck-at := C, plane-at := B, package-at := A  
truck-at := A, plane-at := A, package-at := B  
truck-at := B, plane-at := B, package-at := C  
truck-at := C, plane-at := C, package-at := B  
truck-at := A, plane-at := B, package-at := C  
truck-at := B, plane-at := C, package-at := A  
truck-at := C, plane-at := A, package-at := B  
truck-at := A, plane-at := C, package-at := A  
truck-at := B, plane-at := A, package-at := C  
truck-at := C, plane-at := B, package-at := A

# MPT/FDR

## Example

✓ variables and their domains:

- ▶ truck-at  $\in \{A, B\}$
- ▶ plane-at  $\in \{B, C\}$
- ▶ package-at  $\in \{A, B, C, t, a\}$
- ▶  $2 \times 2 \times 5 = 20$  states

○ operators:

```
truck-at := {A, B}
plane-at := {B, C}
package-at := {A, B, C, t, a}

truck-at := A
truck-at := B
plane-at := B
plane-at := C
package-at := A
package-at := B
package-at := C
package-at := t
package-at := a
```

# MPT/FDR

## Example

✓ variables and their domains:

- ▶ truck-at  $\in \{A, B\}$
- ▶ plane-at  $\in \{B, C\}$
- ▶ package-at  $\in \{A, B, C, t, a\}$
- ▶  $2 \times 2 \times 5 = 20$  states

○ operators:

```

truck-at := {A, B}
plane-at := {B, C}
package-at := {A, B, C, t, a}

truck-at := A
truck-at := B
plane-at := B
plane-at := C
package-at := A
package-at := B
package-at := C
package-at := t
package-at := a

```

# MPT/FDR

## Example

✓ variables and their domains:

- ▶ truck-at  $\in \{A, B\}$
- ▶ plane-at  $\in \{B, C\}$
- ▶ package-at  $\in \{A, B, C, t, a\}$
- ▶  $2 \times 2 \times 5 = 20$  states

○ operators:

truck-at(A) → truck-at(B)  
truck-at(B) → truck-at(A)  
plane-at(B) → plane-at(C)  
plane-at(C) → plane-at(B)  
package-at(A) → package-at(t)  
package-at(t) → package-at(A)  
package-at(B) → package-at(a)  
package-at(a) → package-at(B)  
package-at(C) → package-at(a)  
package-at(a) → package-at(C)

# MPT/FDR

## Example

✓ variables and their domains:

- ▶ truck-at  $\in \{A, B\}$
- ▶ plane-at  $\in \{B, C\}$
- ▶ package-at  $\in \{A, B, C, t, a\}$
- ▶  $2 \times 2 \times 5 = 20$  states

○ operators:

# MPT/FDR

## Example

✓ variables and their domains:

- ▶ truck-at  $\in \{A, B\}$
- ▶ plane-at  $\in \{B, C\}$
- ▶ package-at  $\in \{A, B, C, t, a\}$
- ▶  $2 \times 2 \times 5 = 20$  states

○ operators:



# MPT/FDR

## Example

✓ variables and their domains:

- ▶ truck-at  $\in \{A, B\}$
- ▶ plane-at  $\in \{B, C\}$
- ▶ package-at  $\in \{A, B, C, t, a\}$
- ▶  $2 \times 2 \times 5 = 20$  states

○ operators:

load-p-a-B: pre: plane-at=B, package-at=B  
eff: package-at=a

move-t-A-B: pre: truck-at=A  
eff: truck-at=B

# MPT/FDR

## Example

∨ variables and their domains:

- ▶ truck-at  $\in \{A, B\}$
- ▶ plane-at  $\in \{B, C\}$
- ▶ package-at  $\in \{A, B, C, t, a\}$
- ▶  $2 \times 2 \times 5 = 20$  states

○ operators:

**load-p-a-B:** pre: plane-at=B, package-at=B  
eff: package-at=a

**move-t-A-B:** pre: truck-at=A  
eff: truck-at=B

# MPT/FDR

## Example

∨ variables and their domains:

- ▶ truck-at  $\in \{A, B\}$
- ▶ plane-at  $\in \{B, C\}$
- ▶ package-at  $\in \{A, B, C, t, a\}$
- ▶  $2 \times 2 \times 5 = 20$  states

○ operators:

- load-p-a-B:** pre: plane-at=B, package-at=B  
eff: package-at=a
- move-t-A-B:** pre: truck-at=A  
eff: truck-at=B

# PDDL

## Planning Domain Definition Language

- ▶ General language (predicate logic) to describe planning problems

**Domain file** definition of types, predicates, operators

**Problem file** definition of objects, initial state and goal

- ▶ Lisp-like syntax
- ▶ Prefix notation (+1 2)
- ▶ A lot of brackets
- ▶ Several versions (1.2, 2.1, 3.1)

# PDDL

## Planning Domain Definition Language

- ▶ General language (predicate logic) to describe planning problems

**Domain file** definition of types, predicates, operators

**Problem file** definition of objects, initial state and goal

- ▶ Lisp-like syntax
- ▶ Prefix notation (+1 2)
- ▶ A lot of brackets
- ▶ Several versions (1.2, 2.1, 3.1)

# PDDL

## Planning Domain Definition Language

- ▶ General language (predicate logic) to describe planning problems

**Domain file** definition of types, predicates, operators

**Problem file** definition of objects, initial state and goal

- ▶ Lisp-like syntax
- ▶ Prefix notation (+1 2)
- ▶ A lot of brackets
- ▶ Several versions (1.2, 2.1, 3.1)

# PDDL

## Planning Domain Definition Language

- ▶ General language (predicate logic) to describe planning problems

**Domain file** definition of types, predicates, operators

**Problem file** definition of objects, initial state and goal

- ▶ Lisp-like syntax
- ▶ Prefix notation (+1 2)
- ▶ A lot of brackets
- ▶ Several versions (1.2, 2.1, 3.1)

# PDDL

## Planning Domain Definition Language

- ▶ General language (predicate logic) to describe planning problems

**Domain file** definition of types, predicates, operators

**Problem file** definition of objects, initial state and goal

- ▶ Lisp-like syntax
- ▶ Prefix notation (+12)
- ▶ A lot of brackets
- ▶ Several versions (1.2, 2.1, 3.1)



# PDDL

## Planning Domain Definition Language

- ▶ General language (predicate logic) to describe planning problems

**Domain file** definition of types, predicates, operators

**Problem file** definition of objects, initial state and goal

- ▶ Lisp-like syntax
- ▶ Prefix notation (+1 2)
- ▶ A lot of brackets
- ▶ Several versions (1.2, 2.1, 3.1)

# PDDL

## Planning Domain Definition Language

- ▶ General language (predicate logic) to describe planning problems

**Domain file** definition of types, predicates, operators

**Problem file** definition of objects, initial state and goal

- ▶ Lisp-like syntax
- ▶ Prefix notation (+1 2)
- ▶ A lot of brackets
- ▶ Several versions (1.2, 2.1, 3.1)

# PDDL

## Planning Domain Definition Language

- ▶ General language (predicate logic) to describe planning problems

**Domain file** definition of types, predicates, operators

**Problem file** definition of objects, initial state and goal

- ▶ Lisp-like syntax
- ▶ Prefix notation (+1 2)
- ▶ A lot of brackets
- ▶ Several versions (1.2, 2.1, 3.1)

# PDDL

## Grounding

### ▶ Predicate logic

#### ▶ → STRIPS (propositional logic)

- ▶ All possible instantiations of predicates → propositions
- ▶ All possible instantiations of actions

What if necessary?

What Reachability analysis can help

#### ▶ → FDR (finite domain representation)

→ Grounding, eg. in STRIPS

→ Reachability analysis → PDDL and variants

# PDDL

## Grounding

### ▶ Predicate logic

#### ▶ → STRIPS (propositional logic)

- ▶ All possible instantiations of predicates → propositions
- ▶ All possible instantiations of actions

▶ Is it all necessary?

▶ No! Reachability analysis can help

#### ▶ → FDR (finite domain representation)

▶ Example: eg in STRIPS

▶ Reachability analysis → not always well defined



# PDDL

## Grounding

### ▶ Predicate logic

#### ▶ → STRIPS (propositional logic)

- ▶ All possible instantiations of predicates → propositions
- ▶ All possible instantiations of actions

▶ Is it all necessary?

▶ No! Reachability analysis can help

#### ▶ → FDR (finite domain representation)

▶ Example: eg. In STRIPS

▶ `has(robot, location) :- has(robot, location) && has(robot, location)`





# PDDL

## Grounding

- ▶ Predicate logic

- ▶ → STRIPS (propositional logic)

- ▶ All possible instantiations of predicates → propositions

- ▶ All possible instantiations of actions

- ▶ Is it all necessary?

- ▶ No! Reachability analysis can help

- ▶ → FDR (finite domain representation)

- ▶ Grounding is like STRIPS

- ▶ Grounding is not needed for PDDL solvers

# PDDL

## Grounding

- ▶ Predicate logic

- ▶ → STRIPS (propositional logic)

- ▶ All possible instantiations of predicates → propositions

- ▶ All possible instantiations of actions

- ▶ Is it all necessary?

- ▶ No! Reachability analysis can help

- ▶ → FDR (finite domain representation)

- ▶ Grounding is like STRIPS

- ▶ Grounding is not needed for PDDL solvers

# PDDL

## Grounding

- ▶ Predicate logic

- ▶ → STRIPS (propositional logic)

- ▶ All possible instantiations of predicates → propositions

- ▶ All possible instantiations of actions

- ▶ Is it all necessary?

- ▶ No! Reachability analysis can help

- ▶ → FDR (finite domain representation)

# PDDL

## Grounding

- ▶ Predicate logic

- ▶ → STRIPS (propositional logic)

- ▶ All possible instantiations of predicates → propositions

- ▶ All possible instantiations of actions

- ▶ Is it all necessary?

- ▶ No! Reachability analysis can help

- ▶ → FDR (finite domain representation)

# PDDL

## Grounding

- ▶ Predicate logic
  - ▶ → STRIPS (propositional logic)
    - ▶ All possible instantiations of predicates → propositions
    - ▶ All possible instantiations of actions
    - ▶ Is it all necessary?
    - ▶ No! Reachability analysis can help
  - ▶ → FDR (finite domain representation)
    - ▶ Grounding as in STRIPS
    - ▶ Invariant analysis - variables and domains

# PDDL

## Grounding

- ▶ Predicate logic
  - ▶ → STRIPS (propositional logic)
    - ▶ All possible instantiations of predicates → propositions
    - ▶ All possible instantiations of actions
    - ▶ Is it all necessary?
    - ▶ No! Reachability analysis can help
  - ▶ → FDR (finite domain representation)
    - ▶ Grounding as in STRIPS
    - ▶ Invariant analysis - variables and domains

# PDDL

## Grounding

- ▶ Predicate logic
  - ▶ → STRIPS (propositional logic)
    - ▶ All possible instantiations of predicates → propositions
    - ▶ All possible instantiations of actions
    - ▶ Is it all necessary?
    - ▶ No! Reachability analysis can help
  - ▶ → FDR (finite domain representation)
    - ▶ Grounding as in STRIPS
    - ▶ Invariant analysis - variables and domains

# PDDL

## Resources

- ▶ Online editor:  
<http://editor.planning.domains>
- ▶ Resources:  
<http://ipc.informatik.uni-freiburg.de/PddlResources>  
<http://users.cecs.anu.edu.au/~patrik/pddlman/writing.html>  
<http://www.cs.toronto.edu/~sheila/2542/f10/A1/introtopddl2.pdf>  
[http://en.wikipedia.org/wiki/Planning\\_Domain\\_Definition\\_Language](http://en.wikipedia.org/wiki/Planning_Domain_Definition_Language)



# PDDL

## Resources

- ▶ Online editor:  
<http://editor.planning.domains>
- ▶ Resources:  
<http://ipc.informatik.uni-freiburg.de/PddlResources>  
<http://users.cecs.anu.edu.au/~patrik/pddlman/writing.html>  
<http://www.cs.toronto.edu/~sheila/2542/f10/A1/introtopddl2.pdf>  
[http://en.wikipedia.org/wiki/Planning\\_Domain\\_Definition\\_Language](http://en.wikipedia.org/wiki/Planning_Domain_Definition_Language)