

# Knowledge Representation and Reasoning: Planning

Jussi Rintanen

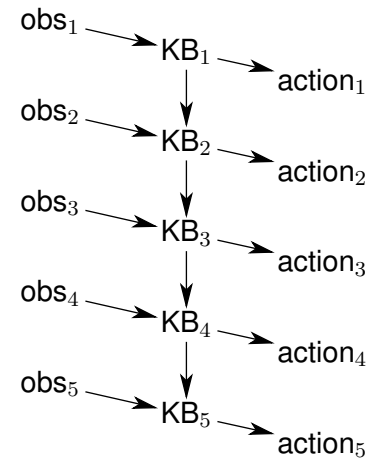
NICTA, Canberra

August 2, 2007

Introduction

1 / 54

## What is planning?



Planning is **decision making** about which actions to take.

- ▶ knowledge base (KB) about the world
- ▶ general-purpose problem representation (PDDL, logic, ...)
- ▶ algorithms for solving any problem expressible in the representation

Introduction

2 / 54

## What is planning?

Application areas:

- ▶ high-level planning for intelligent robots
- ▶ autonomous systems: NASA Deep Space One, ...
- ▶ problem-solving (games like Rubik's cube)
- ▶ production planning
- ▶ related problems: scheduling, time-tabling, ...

3 / 54

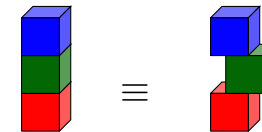
## Blocks world

The states

Location on the table does not matter



Location on a block does not matter



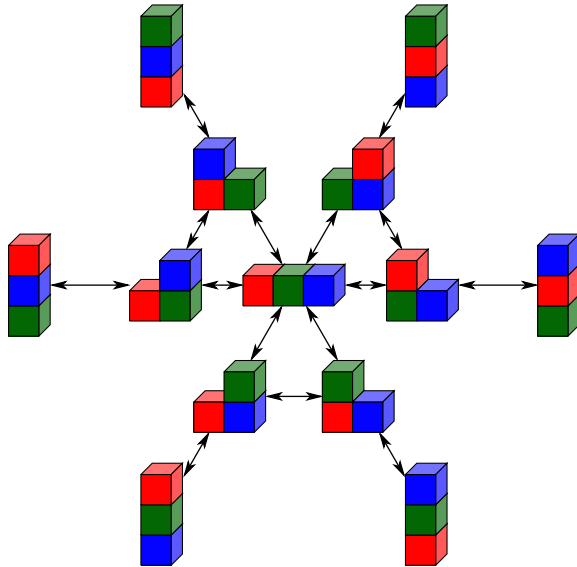
At most one block on/under a block is allowed



4 / 54

## Blocks world

The transition graph for three blocks



## Why is planning difficult?

- ▶ Solutions to simplest planning problems are **paths from an initial state to a goal state** in the transition graph. Efficiently solvable e.g. by Dijkstra's algorithm in  $O(n \log n)$  time.
- ▶ Q: Why don't we solve all planning problems this way?
- ▶ A: State spaces are often huge:  $10^9, 10^{12}, 10^{15}, \dots$  states. Constructing the transition graph explicitly is not feasible!!
- ▶ Planning algorithms often are – but are not guaranteed to be – more efficient than the obvious solution method of constructing the transition graph + running e.g. Dijkstra's algorithm.

## Properties of the world: nondeterminism

### Deterministic world/actions

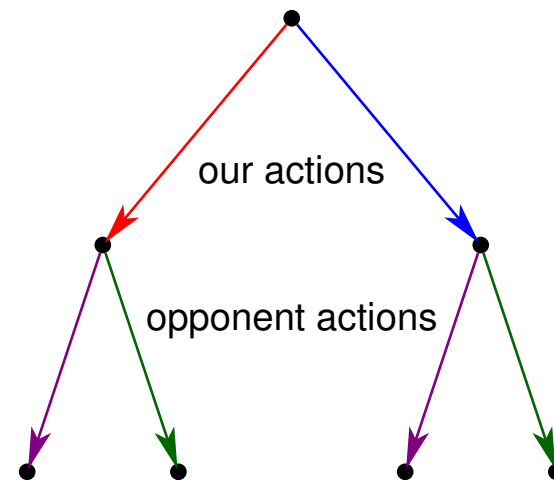
Action and current state **uniquely** determine the successor state.

### Nondeterministic world/actions

For an action and a current state there may be **several successor states**.

## Nondeterminism

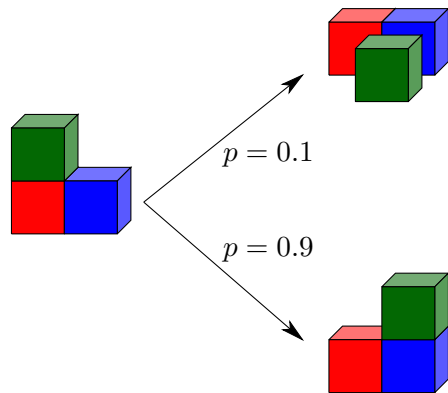
Example: several agents, games



## Nondeterminism

### Example

Moving objects with an unreliable robotic hand: move the green block onto the blue block.



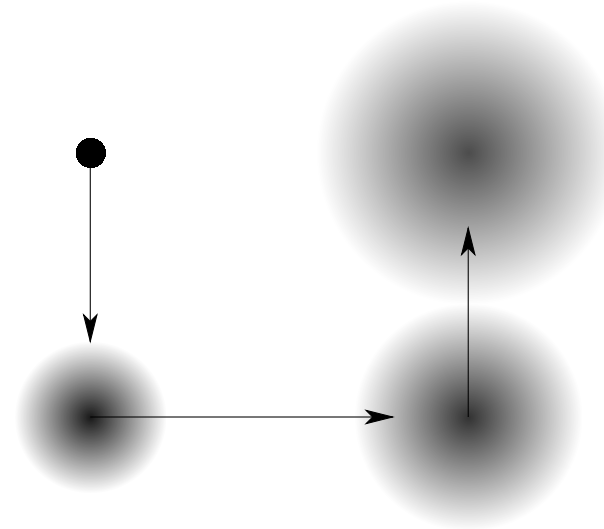
## Nondeterminism

### Motivation

- ▶ World is not predictable.
- ▶ AI robotics:
  - ▶ imprecise movement of the robot
  - ▶ other robots
  - ▶ human beings, animals
  - ▶ machines (cars, trains, airplanes, lawn-mowers, ...)
  - ▶ natural phenomena (wind, water, snow, temperature, ...)
- ▶ Games: other players are outside our control.
- ▶ To win a game (reaching a goal state) with certainty all possible actions by the other players have to be anticipated (a **winning strategy** of a game).
- ▶ World is not predictable because it is unknown: we cannot **observe** everything.

## Nondeterminism

### Example: uncertainty in robot movement



## Nondeterministic

### Motivation

#### Implications:

1. The future state of the world cannot be predicted.
2. We cannot reliably plan ahead: no single action sequence achieves the goals.  
A plan is a more complicated object, similar to a program.
3. In some cases it is not possible to achieve the goals with certainty, only with some probability.

# Properties of the world: observability

## Full observability

Observations/sensing allow to determine the current state of the world **uniquely**.

## Partial observability

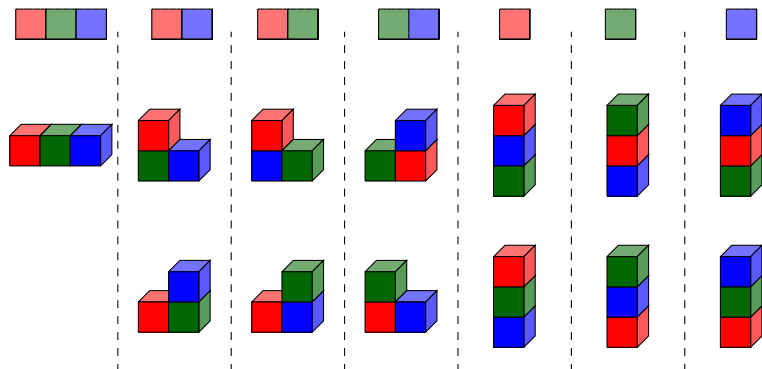
Observations/sensing allow to determine the current state of the world **only partially**: we only know that the current state is one of several of possible ones.

**Consequence**: It is necessary to represent the **knowledge** of an agent.

# Observability

Example of partition of states into observational classes

Blocks world with 3 blocks and a camera high above the table. State variables  $V = \{Aclear, Bclear, Cclear\}$  are observable.

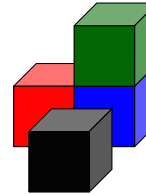


There

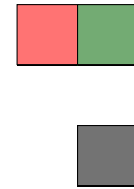
are 8 valuations of  $V$  but the valuation  $v \models \neg Aclear \wedge \neg Bclear \wedge \neg Cclear$  does not correspond to a blocks world state.

# What difference does observability make?

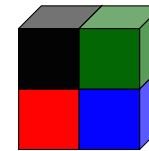
Camera A



Camera B



Goal



# Observability

Classification full, partial, no observability

Let  $S = \{s_1, \dots, s_n\}$  be the set of states.

Classification of planning problems in terms of **observability**:

**Full** Any two states can be distinguished.  
**Chess** is a fully observable 2-person game.

**No** No two states can be distinguished.  
 A robot with all sensors broken.

**Partial** Some states can be distinguished.  
**Poker** is a partially observable 2-person game.  
**Mastermind** is a partially observable 1-person game.

## Different classes of planning problems

actions	deterministic	nondeterministic
probabilities	no	yes
observability	full	partial
horizon	finite	infinite
⋮		

1. classical planning
2. conditional planning with full/partial observability
3. Markov decision processes (MDP)
4. partially observable MDPs (POMDP)

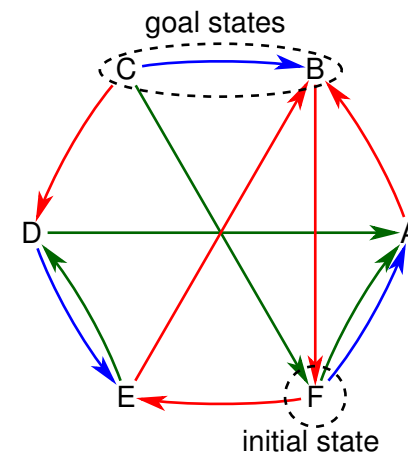
## What do you learn in this lecture?

1. Representation of planning problems
2. Techniques for solving the basic deterministic/classical planning problem:
  - 2.1 algorithms based on heuristic search
  - 2.2 planning by satisfiability testing (SAT)

## Different objectives

1. Reach a goal state.  
**Example:** Earn 500 euro.
2. Stay in goal states indefinitely (infinite horizon).  
**Example:** Never allow the bank account balance to be negative.
3. Maximize the *probability* of reaching a goal state.  
**Example:** Study hard and save money to be able to finance a house by 2015.
4. Collect the maximal *expected* rewards / minimal expected costs (infinite horizon).  
**Example:** Maximize your future income.
5. ...

## Transition systems



## Representation of transition systems

- ▶ state = valuation of a **finite set** of state variables

### Example

HOUR : {0, ..., 23} = 13

MINUTE : {0, ..., 59} = 55

LOCATION : { 51, 52, 82, 101, 102 } = 101

WEATHER : { sunny, cloudy, rainy } = cloudy

HOLIDAY : { T, F } = F

- ▶ Any  $n$ -valued state variable can be represented by  $\lceil \log_2 n \rceil$  Boolean (2-valued) state variables.
- ▶ Actions change the values of the state variables.

## Actions represented as STRIPS operators

### STRIPS operator

A STRIPS operator  $\langle p, e \rangle$  consists of

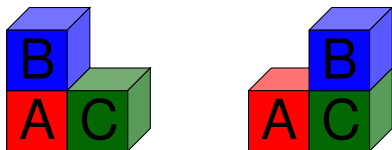
1. a **precondition**  $p$  and
2. an **effect**  $e$

which are both sets of literals.

### Example

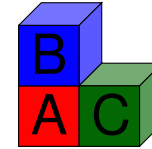
Operator for moving  $B$  from  $A$  to  $C$ :

$\langle \{ \text{BonA}, \text{clearB}, \text{clearC} \}, \{ \text{BonC}, \text{clearA}, \neg \text{BonA}, \neg \text{clearC} \} \rangle$ .



## Blocks world with Boolean state variables

### Example



$$\begin{aligned}
 s(\text{clearA}) &= 0 & s(\text{clearB}) &= 1 & s(\text{clearC}) &= 1 \\
 s(\text{AonB}) &= 0 & s(\text{AonC}) &= 0 & s(\text{AonTABLE}) &= 1 \\
 s(\text{BonA}) &= 1 & s(\text{BonC}) &= 0 & s(\text{BonTABLE}) &= 0 \\
 s(\text{ConA}) &= 0 & s(\text{ConB}) &= 0 & s(\text{ConTABLE}) &= 1
 \end{aligned}$$

Not all valuations correspond to an intended state, e.g. if  $s(\text{AonB}) = 1$  and  $s(\text{BonA}) = 1$ .

## Actions as formulae

### Idea

Propositional variables  $a, b, c, \dots$  for **old** and  $a', b', c', \dots$  for **new** values of state variables.

$$\begin{aligned}
 &(\text{BonA} \wedge \text{clearB} \wedge \text{clearC}) && \text{precondition} \\
 &\wedge (\text{BonC}' \wedge \text{clearA}' \wedge \neg \text{BonA}' \wedge \neg \text{clearC}') && \text{effect} \\
 &\wedge (\text{clearB} \leftrightarrow \text{clearB}') && \text{state variables} \\
 &\wedge (\text{AonB} \leftrightarrow \text{AonB}') && \text{that do not change} \\
 &\wedge (\text{AonC} \leftrightarrow \text{AonC}') && \\
 &\wedge (\text{AonTABLE} \leftrightarrow \text{AonTABLE}') && \\
 &\wedge (\text{BonTABLE} \leftrightarrow \text{BonTABLE}') && \\
 &\wedge (\text{ConA} \leftrightarrow \text{ConA}') && \\
 &\wedge (\text{ConB} \leftrightarrow \text{ConB}') && \\
 &\wedge (\text{ConTABLE} \leftrightarrow \text{ConTABLE}') &&
 \end{aligned}$$

## Sets (of states) as formulas

### Formulas over $A$ represent sets

$a \vee b$  over  $A = \{a, b, c\}$   
 represents the **set**  $\{010, 011, 100, 101, 110, 111\}$ .

### Formulas over $A \cup A'$ represent binary relations

$a \wedge a' \wedge (b \leftrightarrow b')$  over  $A \cup A'$  where  $A = \{a, b\}$ ,  $A' = \{a', b'\}$   
 represents the **binary relation**  $\{(10, 10), (11, 11)\}$ .

Valuations  $1010$  and  $1111$  of  $A \cup A'$  can be viewed respectively as **pairs of valuations**  $(10, 10)$  and  $(11, 11)$  of  $A$ .

## Actions as formulae

### Example

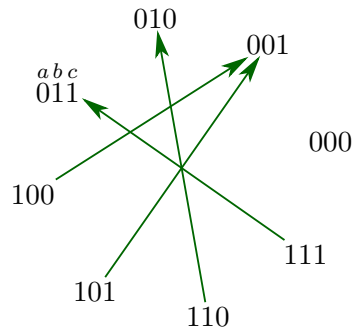
State variables are  $A = \{a, b, c\}$ .

The formula

$$a \wedge \neg a' \wedge (b \leftrightarrow b') \wedge ((\neg b \vee c) \leftrightarrow c')$$

corresponds to the binary relation on the right.

This cannot be expressed as a STRIPS operator, but there are more expressive languages for expressing operators (PDDL).



## Actions as formulae

### From STRIPS operators to formulae

STRIPS operator  $\langle p, e \rangle$  corresponds to the formula

$$\bigwedge_{l \in p} l \wedge \bigwedge_{a \in e \cap A} a' \wedge \bigwedge_{\neg a \in e} \neg a' \wedge \bigwedge_{a \in A \setminus e, \neg a \notin e} (a \leftrightarrow a').$$

Here  $A$  is the set of all state variables.

### General schema for deterministic operators

$p \wedge \bigwedge_{a \in A} (\phi_a \leftrightarrow a')$  where  $\phi_a$  is the new value of  $a$ .

In the STRIPS case:

$\phi_a \leftrightarrow a'$	meaning
$a \leftrightarrow a'$	$a$ does not change
$\top \leftrightarrow a'$	$a$ becomes true
$\perp \leftrightarrow a'$	$a$ becomes false

## Operators

The successor state of a state

### Executability of an operator

$\langle p, e \rangle$  is **executable in a state  $s$**  iff  $s \models p$  and  $e$  is consistent.

### Successor states

The **successor state**  $exec_o(s)$  of  $s$  with respect to  $o = \langle p, e \rangle$  is obtained from  $s$  by making literals in  $e$  true.

This is defined only if  $o$  is executable in  $s$ .

### Example

$\{\{a\}, \{\neg a, b\}\}$  is executable in state  $s$  such that  $s \models a \wedge b \wedge c$  because  $s \models a$  and  $\{\neg a, b\}$  is consistent.

Hence  $exec_{\{\{a\}, \{\neg a, b\}\}}(s) \models \neg a \wedge b \wedge c$ .

## Transition systems

### Transition system $\langle A, I, O, G \rangle$

- ▶  $A$  is a finite set of **state variables**,
- ▶  $I$  is an **initial state** (a valuation of  $A$ ),
- ▶  $O$  is a set of **operators** over  $A$ ,
- ▶  $G$  is a set of literals over  $A$ , the **goals**.

State-space search

## Planning by state-space search

There are many alternative ways of doing planning by state-space search.

1. different ways of expressing planning as a search problem:
  - 1.1 **search direction**: forward, backward
  - 1.2 **representation** of search space: states, sets of states
2. different **search algorithms**:
  - 2.1 depth-first, breadth-first, bidirectional, ...
  - 2.2 heuristic search (**systematic**: A\*, IDA\*, best first, ...; **local**: hill-climbing, simulated annealing, ...), ...
3. different ways of controlling search:
  - 3.1 **different heuristics** for heuristic search algorithms
  - 3.2 **pruning techniques**: invariants, symmetry elimination, ...

29 / 54

31 / 54

## Plans

### Plans

A **plan** for  $\langle A, I, O, G \rangle$  is a sequence  $\pi = o_1, \dots, o_n$  of operators such that  $o_1, \dots, o_n \in O$  and there is a sequence of states  $s_0, \dots, s_n$  (the **execution** of  $\pi$ ) so that

1.  $s_0 = I$ ,
2.  $s_i = \text{exec}_{o_i}(s_{i-1})$  for every  $i \in \{1, \dots, n\}$ , and
3.  $s_n \models G$ .

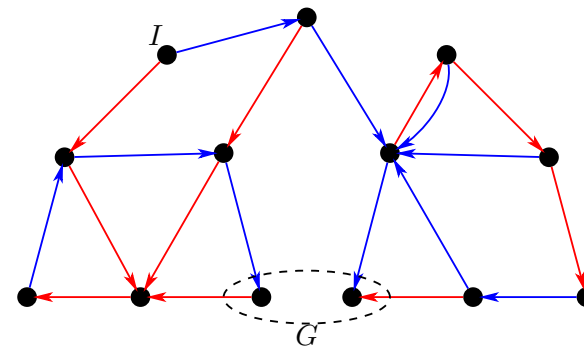
This can be equivalently expressed as

$$\text{exec}_{o_n}(\text{exec}_{o_{n-1}}(\dots \text{exec}_{o_1}(I) \dots)) \models G.$$

State-space search

## Planning by forward search

with depth-first search



30 / 54

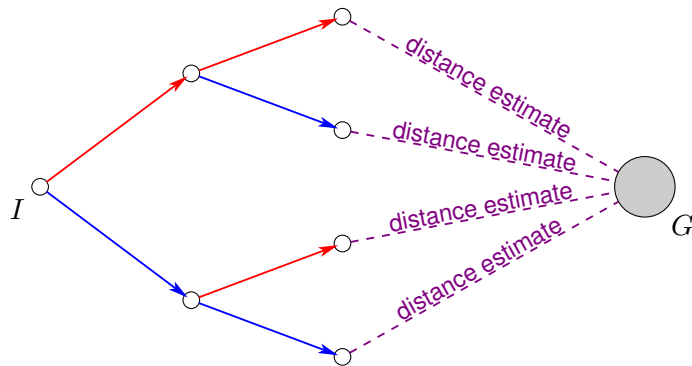
32 / 54

# Progression

- ▶ **Progression** means computing the successor state  $exec_o(s)$  of  $s$  with respect to  $o = \langle p, e \rangle$ . This is simply by making literals in  $e$  true in  $s$ .
- ▶ Used in **forward search**: from the initial state toward the goal states.
- + Very easy and efficient to implement.
- Search with only one state at a time.

# Planning by heuristic search

Forward search



# Regression

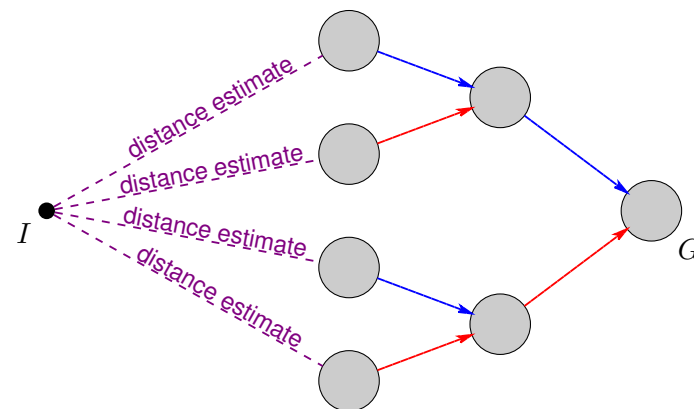
- ▶ **Regression** = computation of predecessors of states

## Regression for STRIPS operators

- ▶ Goals are sets of literals  $\{l_1, \dots, l_n\}$ .
- ▶ **First step**: Choose an operator that makes some of  $l_1, \dots, l_n$  true and makes none of them false.
- ▶ **Second step**: Form a new goal by removing the fulfilled goal literals and adding the preconditions of the operator.

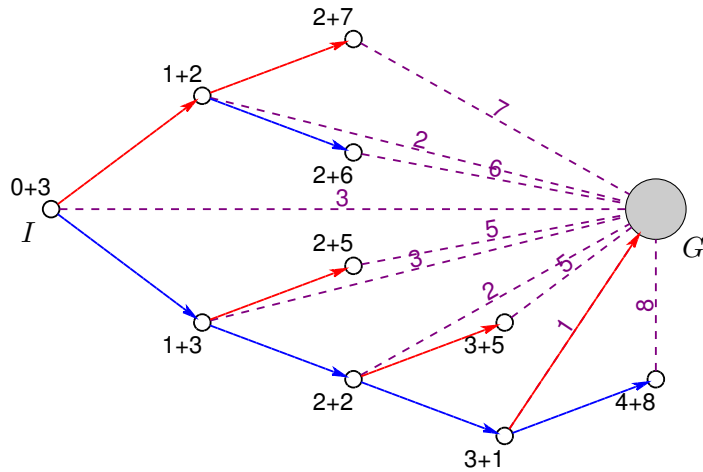
# Planning by heuristic search

Backward search



# Search algorithms: A\*

Example

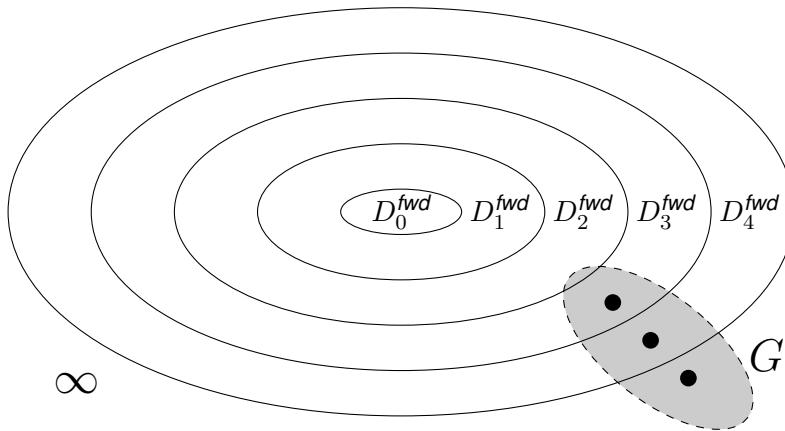


Distances Distances

## Distances

of formulas

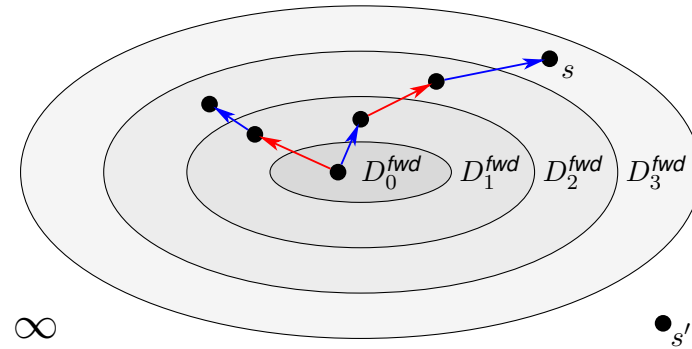
$\delta_I^{fwd}(G) = 3$  since  $s \models G$  for some  $s \in D_3^{fwd}$  and for no  $s \in D_2^{fwd}$ .



## Distances

Illustration

Forward distance of state  $s$  is 3 because  $s \in D_3^{fwd} \setminus D_2^{fwd}$ .



As  $D_i^{fwd} = D_3^{fwd}$  for all  $i > 3$ , forward distance of state  $s'$  is  $\infty$ .

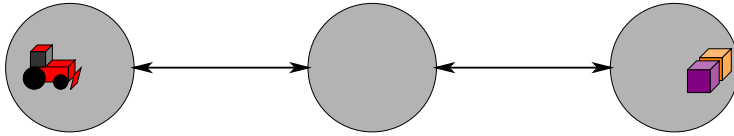
Distances Illustration

## Distance estimation

- ▶ Computation of exact distances is as hard as planning itself: only their approximations are useful as heuristics.
- ▶ We discuss a distance heuristic for controlling heuristic search algorithms like A\*, IDA\*.
- ▶ The distance estimates are a **lower bound** for forward distances: since they don't overestimate they are **admissible** as a heuristic.
- ▶ They can be used with A\* and IDA\* to find **optimal plans**.
- ▶ Basic insight: estimate distances one state variable at a time.

## Distance estimation

### Tractor example



#### 1. Tractor moves:

- ▶ from 1 to 2:  $T_{12} = \langle \{T1\}, \{T2, \neg T1\} \rangle$
- ▶ from 2 to 1:  $T_{21} = \langle \{T2\}, \{T1, \neg T2\} \rangle$
- ▶ from 2 to 3:  $T_{23} = \langle \{T2\}, \{T3, \neg T2\} \rangle$
- ▶ from 3 to 2:  $T_{32} = \langle \{T3\}, \{T2, \neg T3\} \rangle$

#### 2. Tractor pushes A:

- ▶ from 2 to 1:  $A_{21} = \langle \{T2, A2\}, \{T1, A1, \neg T2, \neg A2\} \rangle$
- ▶ from 3 to 2:  $A_{32} = \langle \{T3, A3\}, \{T2, A2, \neg T3, \neg A3\} \rangle$

#### 3. Tractor pushes B:

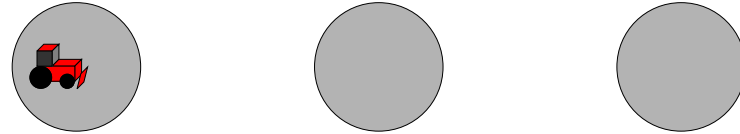
- ▶ from 2 to 1:  $B_{21} = \langle \{T2, B2\}, \{T1, B1, \neg T2, \neg B2\} \rangle$
- ▶ from 3 to 2:  $B_{32} = \langle \{T3, B3\}, \{T2, B2, \neg T3, \neg B3\} \rangle$

## Planning in the propositional logic

- ▶ Planning by **satisfiability testing** in the propositional logic: A planning problem is translated into a formula with parameter  $t$  that is satisfiable if and only if a plan of a length  $t$  exists.
- ▶ Benefits:
  1. Upper bound  $t$  constrains the set of possible plans very strongly, which often makes plan search much easier.
  2. There are very efficient algorithm implementations for satisfiability: zChaff, MiniSAT, ...

## Distance estimation

### Tractor example



$t$	T1	T2	T3	A1	A2	A3	B1	B2	B3
0	T	F	F	F	F	T	F	F	T
1	TF	TF	F	F	F	T	F	F	T
2	TF	TF	TF	F	F	T	F	F	T
3	TF	TF	TF	F	TF	TF	F	TF	TF
4	TF	TF	TF	TF	TF	TF	TF	TF	TF

Distance of  $A1, B1$  is 4.

## Existence of plans of length $t$

### Atomic propositions

Define  $A^i = \{a^i | a \in A\}$  for all  $i \in \{0, \dots, t\}$ .  
 $a^i$  expresses the value of  $a \in A$  at time  $i$ .

### Plans of length $t$ in the propositional logic

Plans of length  $t$  correspond to satisfying valuations of

$$\Phi_t = \iota^0 \wedge \mathcal{R}_1(A^0, A^1) \wedge \mathcal{R}_1(A^1, A^2) \wedge \dots \wedge \mathcal{R}_1(A^{t-1}, A^t) \wedge G^t$$

where  $\iota^0 = \bigwedge \{a^0 | a \in A, I(a) = 1\} \cup \{\neg a^0 | a \in A, I(a) = 0\}$  and  $G^t$  is  $\bigwedge G$  with propositional variables  $a$  replaced by  $a^t$ .

## Planning as satisfiability

### Transition relation in the propositional logic

For  $\langle A, I, O, G \rangle$  define

$$\mathcal{R}_1(A, A') = \bigvee_{o \in O} \phi_o$$

where  $\phi_o$  is a formula for the STRIPS operator  $o$ .

### Plans of length $t$ in the propositional logic

Plans of length  $t$  correspond to satisfying valuations of

$$\Phi_t = I^0 \wedge \mathcal{R}_1(A^0, A^1) \wedge \mathcal{R}_1(A^1, A^2) \wedge \dots \wedge \mathcal{R}_1(A^{t-1}, A^t) \wedge G^t$$

where  $I^0 = \bigwedge \{a^0 \mid a \in A, I(a) = 1\} \cup \{\neg a^0 \mid a \in A, I(a) = 0\}$  and  $G^t$  is  $\bigwedge G$  with propositional variables  $a$  replaced by  $a^t$ .

## Planning as satisfiability with parallel plans

- ▶ **Efficiency** of satisfiability planning is strongly **dependent on the plan length** because satisfiability algorithms have worst-case exponential runtime in the formula size, and formula sizes are linearly proportional to plan length.
- ▶ Formula sizes can be reduced by allowing **several operators in parallel**.
- ▶ On many problems this leads to big speed-ups.

## Planning as satisfiability

### Example

### Example

Consider

$$\begin{aligned} I &\models a \wedge b \\ G &= \{\neg a, b\} \\ o_1 &= \langle \{a\}, \{\neg a, b\} \rangle \\ o_2 &= \langle \{b\}, \{a, \neg b\} \rangle. \end{aligned}$$

Formula for plans of length 3 is

$$\begin{aligned} &(a^0 \wedge b^0) \wedge \\ &((a^0 \wedge \neg a^1 \wedge b^1) \vee (b^0 \wedge a^1 \wedge \neg b^1)) \wedge \\ &((a^1 \wedge \neg a^2 \wedge b^2) \vee (b^1 \wedge a^2 \wedge \neg b^2)) \wedge \\ &((a^2 \wedge \neg a^3 \wedge b^3) \vee (b^2 \wedge a^3 \wedge \neg b^3)) \wedge \\ &(\neg a^3 \wedge b^3). \end{aligned}$$

This formula is satisfiable because the valuation  $v$  such that  $v \models a^0 \wedge b^0 \wedge \neg a^1 \wedge b^1 \wedge a^2 \wedge \neg b^2 \wedge \neg a^3 \wedge b^3$  satisfies it.

## Interpretation of parallelism

### Example

$\langle \{a\}, \{\neg b\} \rangle$  and  $\langle \{b\}, \{\neg a\} \rangle$  have non-contradictory effects and preconditions.

However, neither operator sequence

1.  $\langle \{b\}, \{\neg a\} \rangle, \langle \{a\}, \{\neg b\} \rangle$  nor
2.  $\langle \{a\}, \{\neg b\} \rangle, \langle \{b\}, \{\neg a\} \rangle$

is executable.

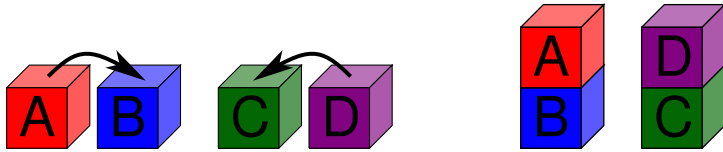
### Requirement for parallel operators

Operators are **executable in every order**.

## Interference

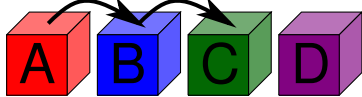
Example

Actions don't interfere



Actions can be taken simultaneously.

Actions interfere



If A is moved first, B won't be clear and cannot be moved.

## Parallel operator execution

Representation in the propositional logic

Let  $o_1, \dots, o_n$  be the operators that make  $a$  true.

Let  $q_1, \dots, q_m$  be the operators that make  $a$  false.

Then the changes in the value of  $a$  are described by the formula

$$C_a = [(o_1 \vee \dots \vee o_n) \vee (a \wedge \neg q_1 \wedge \dots \wedge \neg q_m)] \leftrightarrow a'$$

The change between two time points is described by

$$\mathcal{R}_2(A, A', O) = \bigwedge_{a \in A} C_a \wedge \bigwedge_{\{o_1 \wedge o_2\} \text{ interfere}, o_1 \neq o_2} \neg(o_1 \wedge o_2) \wedge \bigwedge_{(p,e) \in O} o \rightarrow \bigwedge_{l \in p} l'$$

Formula of plans with  $t$  time points is

$$\Phi_t = \iota^0 \wedge \mathcal{R}_2(A^0, A^1, O^0) \wedge \dots \wedge \mathcal{R}_2(A^{t-1}, A^t, O^{t-1}) \wedge G^t.$$

## Interference

Interference

Two operators  $o$  and  $o'$  **interfere** if

1. effects of  $o$  and  $o'$  contradict each other,
2.  $o$  makes the precondition of  $o'$  false, or
3.  $o'$  makes the precondition of  $o$  false.

Example

$\langle \{c\}, \{d\} \rangle$  and  $\langle \{\neg d\}, \{f\} \rangle$  interfere.

$\langle \{c\}, \{d\} \rangle$  and  $\langle \{d\}, \{f\} \rangle$  don't interfere.

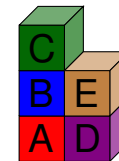
Important property of interference

Any set of **non-interfering** operators that are simultaneously executable in a state  $s$  can be executed in **any order**, leading to the same state in all cases.

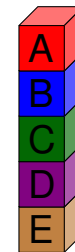
## Planning as satisfiability

Example

initial state



goal state



Problem solved almost without search:

- ▶ Formulas for lengths 1 to 4 shown unsatisfiable without any search.
- ▶ Formula for plan length 5 is satisfiable: 3 nodes in the search tree.
- ▶ Plans have 5 to 7 operators, optimal plan has 5.

# Planning as satisfiability

Example

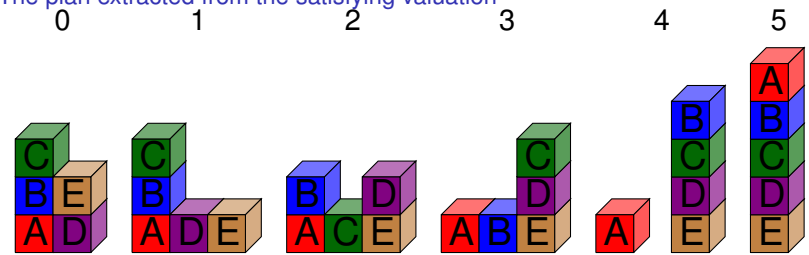
	0	1	2	3	4	5
clear(a)	FF	FFF TT	FFFFTTT			
clear(b)	F F	FF TTF	FFTTF			
clear(c)	TT FF	TTTTFF	TTTTFF			
clear(d)	FTTFFF	FTTFFF	FTTFFF			
clear(e)	TTFFFF	TTFFFF	TTFFFF			
on(a,b)	FFF T	FFFFT	FFFFFT			
on(a,c)	FFFFFF	FFFFFF	FFFFFF			
on(a,d)	FFFFFF	FFFFFF	FFFFFF			
on(a,e)	FFFFFF	FFFFFF	FFFFFF			
on(b,a)	TT FF	TTT FF	TTTTFFF			
on(b,c)	FF TT	FFF TT	FFFFTT			
on(b,d)	FFFFFF	FFFFFF	FFFFFF			
on(b,e)	FFFFFF	FFFFFF	FFFFFF			
on(c,a)	FFFFFF	FFFFFF	FFFFFF			
on(c,b)	T FFF	TT FFF	TTFFF			
on(c,d)	FFFTTT	FFF TTT	FFFFTTT			
on(c,e)	FFFFFF	FFFFFF	FFFFFF			
on(d,a)	FFFFFF	FFFFFF	FFFFFF			
on(d,b)	FFFFFF	FFFFFF	FFFFFF			
on(d,c)	FFFFFF	FFFFFF	FFFFFF			
on(d,e)	FTTTT	FTTTT	FTTTT			
on(e,a)	FFFFFF	FFFFFF	FFFFFF			
on(e,b)	FFFFFF	FFFFFF	FFFFFF			
on(e,c)	FFFFFF	FFFFFF	FFFFFF			
on(e,d)	TFFFFF	TFFFFF	TFFFFF			
ontable(a)	TTT F	TTTTF	TTTTT			
ontable(b)	FF FF	FFF FF	FFFFF			
ontable(c)	F FFF	FF FFF	FFF FFF			
ontable(d)	TTFFFF	TTFFFF	TTFFFF			
ontable(e)	FTTTTT	FTTTTT	FTTTTT			

- State variable values inferred from **initial values** and **goals**.
- Branch:  $\neg\text{clear}(b)^1$ .
- Branch:  $\text{clear}(a)^3$ .
- Plan found:  

	0	1	2	3	4	5
fromtable(a,b)	FFFFT					
fromtable(b,c)	FFFTF					
fromtable(c,d)	FFTFF					
fromtable(d,e)	FTFFF					
totable(b,a)	FTFFF					
totable(c,b)	FTFFF					
totable(e,d)	TFFFF					

# Planning as satisfiability

The plan extracted from the satisfying valuation



Plan with the smallest number of actions:

