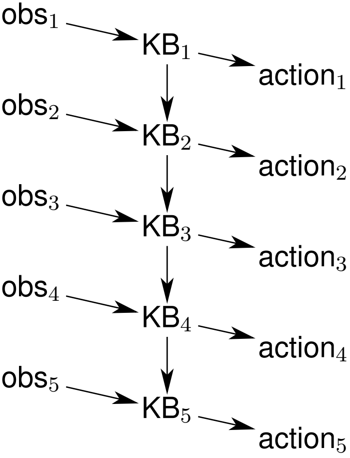
Automated Planning

Jussi Rintanen

NICTA, Canberra

February, 2009

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

What is planning?

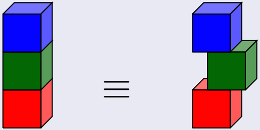
Application areas:

- control of complex technical systems:
  - autonomous spacecraft (NASA Deep Space One)
  - utilities (recovery from electricity network outages)
  - intelligent manufacturing systems
- high-level planning for intelligent robots
- problem-solving (games like Rubik's cube)
- related problems: scheduling, time-tabling, ...

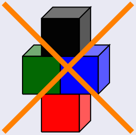
Blocks world
The states

Location on the table does not matter

Location on a block does not matter

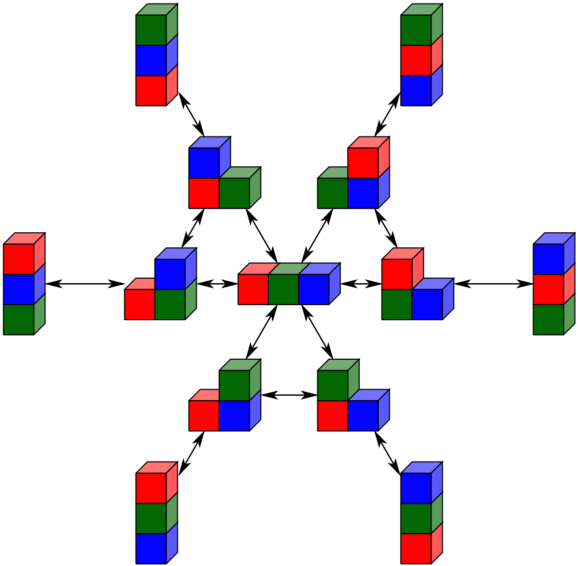


At most one block on/under a block is allowed



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: $10^9, 10^{12}, 10^{15}, \ldots$ 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.

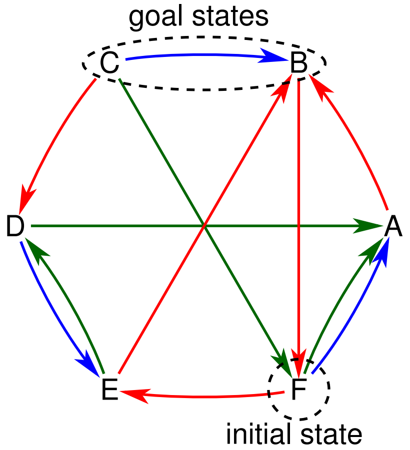
Why is planning difficult?

- 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: $10^9, 10^{12}, 10^{15}, \ldots$ 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.

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: $10^9, 10^{12}, 10^{15}, \ldots$ 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.

Transition systems



Representation of transition systems

- state = valuation of a finite set of state variables

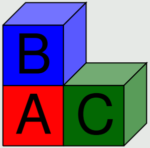
Example

HOUR : $\{0, \dots, 23\}$ = 13
MINUTE : $\{0, \dots, 59\}$= 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.

Blocks world with Boolean state variables

Example



s(clearA)	=	0	s(clearB)	=	1	s(clearC)	=	1
s(AonB)	=	0	s(AonC)	=	0	s(AonTABLE)	=	1
s(BonA)	=	1	s(BonC)	=	0	s(BonTABLE)	=	0
s(ConA)	=	0	s(ConB)	=	0	s(ConTABLE)	=	1

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

Actions

Precondition

A Boolean combination ($\vee$,$\wedge$,$\neg$) of atomic formulas $x = v$ where x is a state variable and v is a value 0 or 1.

Effects

A collection of assignments and conditional assignments

x := v

IF ϕ THEN $x := v$

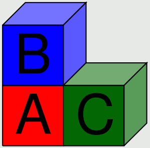
Assumptions:

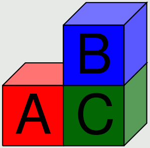
- All assignments in an effect are made simultaneously.
- Only one occurrence of every assignment $x := v$:
  - $x := v$ is equivalent to IF $\top$ THEN $x := v$.
  - Assignments IF ϕ THEN $x := v$ and IF ϕ' THEN $x := v$ can be combined to IF $\phi \vee \phi'$ THEN $x := v$.

We abbreviate $x = 1$ by x and $x = 0$ by $\neg x$, and similarly $x := 1$ by x and $x := 0$ by $\neg x$.

Example

Action for moving B from A to C:
$\langle \text{BonA} \wedge \text{clearB} \wedge \text{clearC}, \{\text{BonC}, \text{clearA}, \neg\text{BonA}, \neg\text{clearC}\}\rangle$.





Active effects of an action

For an action $\langle p, e\rangle$ and state s, $[e]_s$ consists of

$$\begin{cases} x, & \text{for } x := 1 \text{ in } e \\ \neg x, & \text{for } x := 0 \text{ in } e \\ x, & \text{for IF } \phi \text{ THEN } x := 1 \text{ in } e \text{ and } s \models \phi \\ \neg x, & \text{for IF } \phi \text{ THEN } x := 0 \text{ in } e \text{ and } s \models \phi \end{cases}$$

Executability of an action

$\langle p, e\rangle$ is executable in a state s iff $s \models p$ and $[e]_s$ 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 the literals $[e]_s$ true.
This is defined only if o is executable in s.

Example

$\langle a, \{\neg a, b\} \rangle$ 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*$_{\langle a, \{\neg a, b\} \rangle}(s) \models \neg a \wedge b \wedge c$.

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 actions over A,
- G is a formula over A, the goal.

Plans

Plans

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

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

This can be equivalently expressed as

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

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, ...

Actions as formulae

Idea

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

$(\mathsf{BonA} \wedge \mathsf{clearB} \wedge \mathsf{clearC})$	precondition
$\wedge \mathsf{BonC}' \wedge \mathsf{clearA}' \wedge \neg \mathsf{BonA}' \wedge \neg \mathsf{clearC}'$	effects
$\wedge(\mathsf{clearB} \leftrightarrow \mathsf{clearB}')$	state variables
$\wedge(\mathsf{AonB} \leftrightarrow \mathsf{AonB}')$	that do not change
$\wedge(\mathsf{AonC} \leftrightarrow \mathsf{AonC}')$	
$\wedge(\mathsf{AonTABLE} \leftrightarrow \mathsf{AonTABLE}')$	
$\wedge(\mathsf{BonTABLE} \leftrightarrow \mathsf{BonTABLE}')$	
$\wedge(\mathsf{ConA} \leftrightarrow \mathsf{ConA}')$	
$\wedge(\mathsf{ConB} \leftrightarrow \mathsf{ConB}')$	
$\wedge(\mathsf{ConTABLE} \leftrightarrow \mathsf{ConTABLE}')$	

Representation of one event/action

Changes to state variables

effect of e on a	translation $f_e(a)$
-	$a \leftrightarrow a'$
a := 1	a'
a := 0	$\neg a'$
IF ϕ *THEN* a := 1	$(a \vee \phi) \leftrightarrow a'$
IF ϕ *THEN* a := 0	$(a \wedge \neg\phi) \leftrightarrow a'$
IF ϕ_1 *THEN* a := 1; *IF* ϕ_0 *THEN* a := 0	$(\phi_1 \vee (a \wedge \neg\phi_0)) \leftrightarrow a'$

A formula for one event/action e is now defined as

$$F(e) = \phi \wedge \bigwedge_{a \in A} f_e(a)$$

where $\phi = \text{prec}(e)$ is a precondition that has to be true for the event/action to be possible.

Choice between actions/events

A formula that expresses the choice between actions/events $e_1, \dots, e_n$ is

$$\mathcal{R}_1(A, A') = \bigvee_{i=1}^{n} F(e_i).$$

We will later instantiate A and A' with different sets of propositional variables.

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

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

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 G with propositional variables a replaced by a^t.

Example

Consider

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

Formula for plans of length 3 is

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

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.

Parallel plans

- Efficiency of satisfiability testing is strongly dependent on the horizon length because known algorithms have worst-case exponential runtime in the formula size, and formula size is linearly proportional to horizon length.
- Formula sizes can be reduced by allowing several events/actions in parallel.
- On many problems this leads to big speed-ups.

Interpretation of parallelism

Example

$\langle a, \{\neg b\}\rangle$ and $\langle b, \{\neg a\}\rangle$ have non-contradictory effects and preconditions.
However, neither action sequence

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

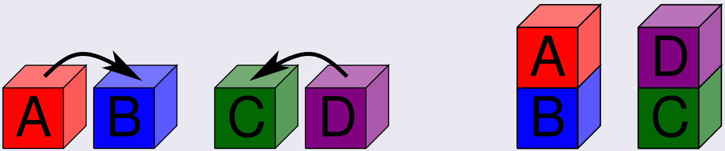
is executable.

Standard interpretation of parallelism

Actions are executable in every order.

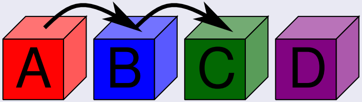
Interference
Example

Actions do not interfere



Actions can be taken simultaneously.

Actions interfere



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

Interference

Interference

o and o' interfere if

1. o may make the precondition of o' false, or
2. o may change the actual effects of o',

or the other way round.

Example

$\langle c, \{d\}\rangle$ and $\langle \neg d, \{f\}\rangle$ interfere.
$\langle c, \{d\}\rangle$ and $\langle d, \{f\}\rangle$ do not interfere.

Important property of interference

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

Formulas $EPC_e^+(a)$ and $EPC_e^-(a)$ indicate when e makes a *true* and *false*, respectively:

$e(a)$	$EPC_e^+(a)$	$EPC_e^-(a)$
-	$\perp$	$\perp$
a := 0	$\perp$	$\top$
a := 1	$\top$	$\perp$
IF ϕ *THEN* a := 1	ϕ	$\perp$
IF ϕ *THEN* a := 0	$\perp$	ϕ
IF ϕ_1 *THEN* a := 1; *IF* ϕ_0 *THEN* a := 0	ϕ_1	ϕ_0

Our earlier definition for effects can be now rephrased as

$$f_e(a) = EPC_e^+(a) \vee (a \wedge \neg EPC_e^-(a)) \leftrightarrow a'.$$

We define $EPC_e^+(a)$ and $EPC_e^-(a)$ as $EPC_e^+(a)$ and

Let $\mathcal{R}_2(A, A', O)$ be the conjunction of

$$(o \rightarrow \text{prec}(o)) \wedge$$
$$\bigwedge_{a \in A} (o \wedge EPC_o^+(a) \rightarrow a') \wedge$$
$$\bigwedge_{a \in A} (o \wedge EPC_o^-(a) \rightarrow \neg a')$$

for all $o \in O$ with effect e and the explanatory frame axioms
$(a \wedge \neg a') \rightarrow ((o_1 \wedge EPC_{o_1}^-(a)) \vee \cdots \vee (o_n \wedge EPC_{o_n}^-(a)))$
$(\neg a \wedge a') \rightarrow ((o_1 \wedge EPC_{o_1}^+(a)) \vee \cdots \vee (o_n \wedge EPC_{o_n}^+(a)))$
for all $a \in A$ where $O = \{o_1, \ldots, o_n\}$, and

$$\bigwedge_{o_1, o_2 \in O \text{ interfere}} \neg(o_1 \wedge o_2).$$

Plans with parallel actions

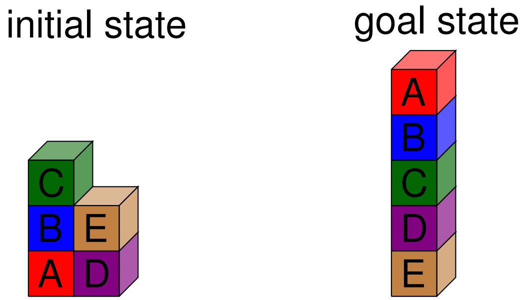
Parallel plans of length n in the propositional logic

Parallel plans of length n correspond to satisfying valuations of

$$I^0 \wedge \mathcal{R}_2(A^0, A^1, O^0) \wedge \cdots \wedge \mathcal{R}_2(A^{n-1}, A^n, O^{n-1}) \wedge G^n$$

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

Planning as satisfiability
Example



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 actions, optimal plan has 5.

Planning as satisfiability
Example

	0	1	2	3	4	5	0	1	2	3	4	5	0	1	2	3	4	5
clear(a)	F	F					F	F	F		T	T	F	F	F	T	T	T
clear(b)	F					F	F	F		T	T	F	F	F	T	T	T	F
clear(c)	T	T			F	F	T	T	T	T	F	F	T	T	T	T	F	F
clear(d)	F	T	T	F	F	F	F	T	T	F	F	F	F	T	T	F	F	F
clear(e)	T	T	F	F	F	F	T	T	F	F	F	F	T	T	F	F	F	F
on(a,b)	F	F	F			T	F	F	F	F	F	T	F	F	F	F	F	T
on(a,c)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(a,d)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(a,e)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(b,a)	T	T			F	F	T	T	T		F	F	T	T	T	F	F	F
on(b,c)	F	F			T	T	F	F	F	F	T	T	F	F	F	F	T	T
on(b,d)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(b,e)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(c,a)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(c,b)	T			F	F	F	T	T		F	F	F	T	T	F	F	F	F
on(c,d)	F	F	F	T	T	T	F	F	F	T	T	T	F	F	F	T	T	T
on(c,e)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(d,a)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(d,b)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(d,c)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(d,e)	F	F	T	T	T	T	F	F	T	T	T	T	F	F	T	T	T	T
on(e,a)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(e,b)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(e,c)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(e,d)	T	F	F	F	F	F	T	F	F	F	F	F	T	F	F	F	F	F
ontable(a)	T	T	T			F	T	T	T	T	T	F	T	T	T	T	T	F
ontable(b)	F	F			F	F	F	F	F		F	F	F	F	F	T	F	F
ontable(c)	F			F	F	F	F	F		F	F	F	F	F	T	F	F	F
ontable(d)	T	T	F	F	F	F	T	T	F	F	F	F	T	T	F	F	F	F
ontable(e)	F	T	T	T	T	T	F	T	T	T	T	T	F	T	T	T	T	T

1. State variable values inferred from initial values and goals.
2. Branch: $\neg$clear(b)[1].
3. Branch: clear(a)[3].
4. Plan found:

	0	1	2	3	4
fromtable(a,b)	F	F	F	F	T
fromtable(b,c)	F	F	F	T	F
fromtable(c,d)	F	F	T	F	F
fromtable(d,e)	F	T	F	F	F
totable(b,a)	F	F	T	F	F
totable(c,b)	F	T	F	F	F
totable(e,d)	T	F	F	F	F

Planning as satisfiability
Example

	0	1	2	3	4	5	0	1	2	3	4	5	0	1	2	3	4	5
clear(a)	F	F					F	F	F		T	T	F	F	F	T	T	T
clear(b)	F					F	F	F		T	T	F	F	F	T	T	T	F
clear(c)	T	T			F	F	T	T	T	T	F	F	T	T	T	T	F	F
clear(d)	F	T	T	F	F	F	F	T	T	F	F	F	F	T	T	F	F	F
clear(e)	T	T	F	F	F	F	T	T	F	F	F	F	T	T	F	F	F	F
on(a,b)	F	F	F			T	F	F	F	F	F	T	F	F	F	F	F	T
on(a,c)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(a,d)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(a,e)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(b,a)	T	T			F	F	T	T	T		F	F	T	T	T	F	F	F
on(b,c)	F	F			T	T	F	F	F	F	T	T	F	F	F	F	T	T
on(b,d)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(b,e)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(c,a)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(c,b)	T			F	F	F	T	T		F	F	F	T	T	F	F	F	F
on(c,d)	F	F	F	T	T	T	F	F	F	T	T	T	F	F	F	T	T	T
on(c,e)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(d,a)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(d,b)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(d,c)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(d,e)	F	F	T	T	T	T	F	F	T	T	T	T	F	F	T	T	T	T
on(e,a)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(e,b)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(e,c)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(e,d)	T	F	F	F	F	F	T	F	F	F	F	F	T	F	F	F	F	F
ontable(a)	T	T	T			F	T	T	T	T	T	F	T	T	T	T	T	F
ontable(b)	F	F			F	F	F	F	F		F	F	F	F	F	T	F	F
ontable(c)	F			F	F	F	F	F		F	F	F	F	F	T	F	F	F
ontable(d)	T	T	F	F	F	F	T	T	F	F	F	F	T	T	F	F	F	F
ontable(e)	F	T	T	T	T	T	F	T	T	T	T	T	F	T	T	T	T	T

1. State variable values inferred from initial values and goals.
2. Branch: $\neg$clear(b)^1.
3. Branch: clear(a)^3.
4. Plan found:

	0	1	2	3	4
fromtable(a,b)	F	F	F	F	T
fromtable(b,c)	F	F	F	T	F
fromtable(c,d)	F	F	T	F	F
fromtable(d,e)	F	T	F	F	F
totable(b,a)	F	F	T	F	F
totable(c,b)	F	T	F	F	F
totable(e,d)	T	F	F	F	F

Planning as satisfiability

Example

	0	1	2	3	4	5	0	1	2	3	4	5	0	1	2	3	4	5
clear(a)	F	F					F	F	F		T	T	F	F	F	T	T	T
clear(b)	F					F	F	F		T	T	F	F	F	T	T	T	F
clear(c)	T	T			F	F	T	T	T	T	F	F	T	T	T	T	F	F
clear(d)	F	T	T	F	F	F	F	T	T	F	F	F	F	T	T	F	F	F
clear(e)	T	T	F	F	F	F	T	T	F	F	F	F	T	T	F	F	F	F
on(a,b)	F	F	F			T	F	F	F	F	F	T	F	F	F	F	F	T
on(a,c)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(a,d)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(a,e)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(b,a)	T	T			F	F	T	T	T		F	F	T	T	T	F	F	F
on(b,c)	F	F			T	T	F	F	F	F	T	T	F	F	F	F	T	T
on(b,d)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(b,e)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(c,a)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(c,b)	T			F	F	F	T	T		F	F	F	T	T	F	F	F	F
on(c,d)	F	F	F	T	T	T	F	F	F	T	T	T	F	F	F	T	T	T
on(c,e)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(d,a)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(d,b)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(d,c)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(d,e)	F	F	T	T	T	T	F	F	T	T	T	T	F	F	T	T	T	T
on(e,a)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(e,b)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(e,c)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(e,d)	T	F	F	F	F	F	T	F	F	F	F	F	T	F	F	F	F	F
ontable(a)	T	T	T			F	T	T	T	T	T	F	T	T	T	T	T	F
ontable(b)	F	F			F	F	F	F	F		F	F	F	F	F	T	F	F
ontable(c)	F			F	F	F	F	F		F	F	F	F	F	T	F	F	F
ontable(d)	T	T	F	F	F	F	T	T	F	F	F	F	T	T	F	F	F	F
ontable(e)	F	T	T	T	T	T	F	T	T	T	T	T	F	T	T	T	T	T

1. State variable values inferred from initial values and goals.
2. Branch: $\neg$clear(b)^1.
3. Branch: clear(a)^3.
4. Plan found:

	0	1	2	3	4
fromtable(a,b)	F	F	F	F	T
fromtable(b,c)	F	F	F	T	F
fromtable(c,d)	F	F	T	F	F
fromtable(d,e)	F	T	F	F	F
totable(b,a)	F	F	T	F	F
totable(c,b)	F	T	F	F	F
totable(e,d)	T	F	F	F	F

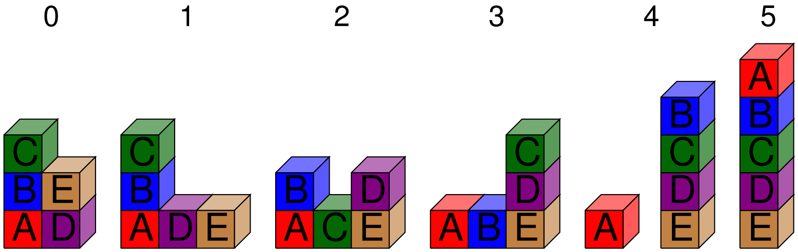
Planning as satisfiability
Example

	0	1	2	3	4	5	0	1	2	3	4	5	0	1	2	3	4	5
clear(a)	F	F					F	F	F		T	T	F	F	F	T	T	T
clear(b)	F					F	F	F		T	T	F	F	F	T	T	T	F
clear(c)	T	T			F	F	T	T	T	T	F	F	T	T	T	T	F	F
clear(d)	F	T	T	F	F	F	F	T	T	F	F	F	F	T	T	F	F	F
clear(e)	T	T	F	F	F	F	T	T	F	F	F	F	T	T	F	F	F	F
on(a,b)	F	F	F			T	F	F	F	F	F	T	F	F	F	F	F	T
on(a,c)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(a,d)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(a,e)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(b,a)	T	T			F	F	T	T	T		F	F	T	T	T	F	F	F
on(b,c)	F	F			T	T	F	F	F	F	T	T	F	F	F	F	T	T
on(b,d)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(b,e)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(c,a)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(c,b)	T			F	F	F	T	T		F	F	F	T	T	F	F	F	F
on(c,d)	F	F	F	T	T	T	F	F	F	T	T	T	F	F	F	T	T	T
on(c,e)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(d,a)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(d,b)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(d,c)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(d,e)	F	F	T	T	T	T	F	F	T	T	T	T	F	F	T	T	T	T
on(e,a)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(e,b)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(e,c)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(e,d)	T	F	F	F	F	F	T	F	F	F	F	F	T	F	F	F	F	F
ontable(a)	T	T	T			F	T	T	T	T	T	F	T	T	T	T	T	F
ontable(b)	F	F			F	F	F	F	F		F	F	F	F	F	T	F	F
ontable(c)	F			F	F	F	F	F		F	F	F	F	F	T	F	F	F
ontable(d)	T	T	F	F	F	F	T	T	F	F	F	F	T	T	F	F	F	F
ontable(e)	F	T	T	T	T	T	F	T	T	T	T	T	F	T	T	T	T	T

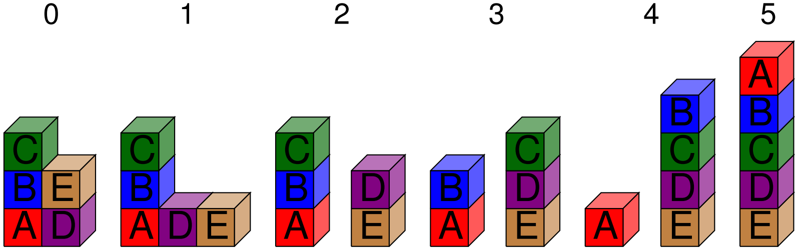
1. State variable values inferred from initial values and goals.
2. Branch: $\neg$clear(b)^1.
3. Branch: clear(a)^3.
4. Plan found:

	0	1	2	3	4
fromtable(a,b)	F	F	F	F	T
fromtable(b,c)	F	F	F	T	F
fromtable(c,d)	F	F	T	F	F
fromtable(d,e)	F	T	F	F	F
totable(b,a)	F	F	T	F	F
totable(c,b)	F	T	F	F	F
totable(e,d)	T	F	F	F	F

Planning as satisfiability
The plan extracted from the satisfying valuation



Plan with the smallest number of actions:



Sets of states as formulas

Formulas over A represent sets of states

A formula ϕ over A can be viewed as representing *all states (valuations of A) that satisfy ϕ.*

Example

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

Relations as formulas

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 $\{(\overset{a\,b}{10}, \overset{a'b'}{10}), (11, 11)\}$.

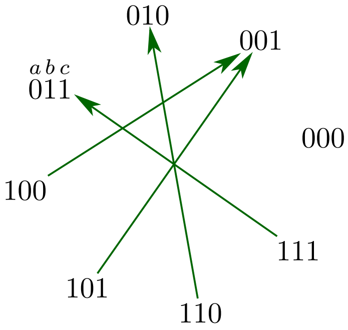
Valuations $\overset{a\,b\,a'b'}{1010}$ and 1111 of $A \cup A'$ can be viewed respectively as pairs of valuations $(\overset{a\,b}{10}, \overset{a'b'}{10})$ and $(11, 11)$ of A.

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.



Relation operations

Join

Join corresponds to Conjunction

Let ϕ_1 and ϕ_2 represent two relations.
Then $\phi_1 \wedge \phi_2$ represents their join.

Example

$b^0 \wedge a^1 \wedge b^1$ $\quad$ $a^1 \wedge (a^2 \vee b^2)$

$$\begin{array}{|c|c|}\hline 01 & 11 \\ 11 & 11 \\ \hline\end{array} \times \begin{array}{|c|c|}\hline 10 & 10 \\ 10 & 01 \\ 10 & 11 \\ 11 & 10 \\ 11 & 01 \\ 11 & 11 \\ \hline\end{array} = \begin{array}{|c|c|c|}\hline 01 & 11 & 10 \\ 01 & 11 & 01 \\ 01 & 11 & 11 \\ 11 & 11 & 10 \\ 11 & 11 & 01 \\ 11 & 11 & 11 \\ \hline\end{array}$$

which is $(b^0 \wedge a^1 \wedge b^1) \wedge (a^1 \wedge (a^2 \vee b^2))$.

Relation operations
Union

Union corresponds to Disjunction

Let ϕ_1 and ϕ_2 represent two relations.
Then $\phi_1 \vee \phi_2$ represents their union.

Example

Union of the two relations represented by the formulas
$a^0 \wedge b^0 \wedge (a^1 \vee b^1)$ $\quad$ $(a^0 \vee \neg b^0) \wedge \neg a^1 \wedge \neg b^1$

$$\begin{array}{|c|c|} \hline 11 & 01 \\ 11 & 10 \\ 11 & 11 \\ \hline \end{array} \quad \cup \quad \begin{array}{|c|c|} \hline 01 & 00 \\ 10 & 00 \\ 11 & 00 \\ \hline \end{array} \quad = \quad \begin{array}{|c|c|} \hline 01 & 00 \\ 10 & 00 \\ 11 & 00 \\ 11 & 10 \\ 11 & 01 \\ 11 & 11 \\ \hline \end{array}$$

corresponds to the formula
$(a^0 \wedge b^0) \vee ((a^0 \vee b^0) \wedge (\neg a^1 \wedge \neg b^1))$.

Relational projection corresponds to the *Abstraction* operation

Let ϕ, on variables A, represent a relation. Let $A' \subseteq A$ represent some columns in the relation.
The projection of the relation to A' is represented by

$$\exists R.\phi$$

where $R = A \backslash A'$. Here $\exists R$ is the existential abstraction operation which will be defined on the next slides.

Existential and universal abstraction

Definition

Existential abstraction of a formula ϕ with respect to $a \in A$:

$$\exists a.\phi = \phi[\top/a] \vee \phi[\bot/a].$$

Universal abstraction is defined analogously by using conjunction instead of disjunction.

Definition

Universal abstraction of a formula ϕ with respect to $a \in A$:

$$\forall a.\phi = \phi[\top/a] \wedge \phi[\bot/a].$$

Example

$$\begin{aligned}
&\exists b.((a \rightarrow b) \wedge (b \rightarrow c)) \\
&= ((a \rightarrow \top) \wedge (\top \rightarrow c)) \vee ((a \rightarrow \bot) \wedge (\bot \rightarrow c)) \\
&\equiv c \vee \neg a \\
&\equiv a \rightarrow c
\end{aligned}$$

$$\begin{aligned}
&\exists ab.(a \vee b) = \exists b.(\top \vee b) \vee (\bot \vee b) \\
&= ((\top \vee \top) \vee (\bot \vee \top)) \vee ((\top \vee \bot) \vee (\bot \vee \bot)) \\
&\equiv (\top \vee \top) \vee (\top \vee \bot) \equiv \top
\end{aligned}$$

Example

∃-abstraction is also known as forgetting:

$\exists \text{mon} \exists \text{tue}((\text{mon} \vee \text{tue}) \wedge (\text{mon} \rightarrow \text{work}) \wedge (\text{tue} \rightarrow \text{work}))$

$\equiv \exists \text{tue}((\text{work} \wedge (\text{tue} \rightarrow \text{work})) \vee (\text{tue} \wedge (\text{tue} \rightarrow \text{work}))) \ \equiv \text{work}$

$\forall$ and $\exists$-abstraction in terms of truth-tables
Example

$\forall c$ and $\exists c$ correspond to combining pairs of lines with the same valuation for variables other than c.

Example

$$\exists c.(a \vee (b \wedge c)) \equiv a \vee b \qquad \forall c.(a \vee (b \wedge c)) \equiv a$$

a	b	c	$a \vee (b \wedge c)$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

a	b	$\exists c.(a \vee (b \wedge c))$
0	0	0
0	1	1
1	0	1
1	1	1

a	b	$\forall c.(a \vee (b \wedge c))$
0	0	0
0	1	0
1	0	1
1	1	1

Applications of $\exists$

The relational operations are important for

- computing immediate predecessors of a set of states,
- computing immediate successors of a set of states,
- all states that are reachable from a set of states.

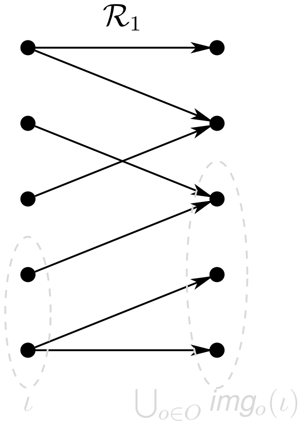
Symbolic reachability computation

$$\iota^0 \wedge \mathcal{R}_1(A^0, A^1) \wedge \cdots \wedge \mathcal{R}_1(A^{t-1}, A^t) \wedge G^t$$

Symbolic reachability computation

$$\mathcal{R}_1(A^0, A^1)$$

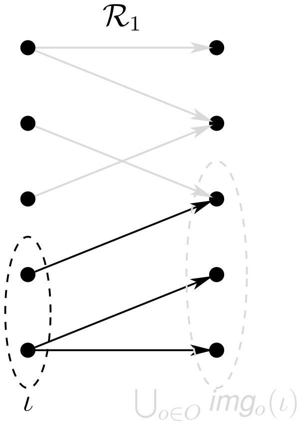
binary relation $\mathcal{R}_1$



Symbolic reachability computation

$$\iota^0 \wedge \mathcal{R}_1(A^0, A^1)$$

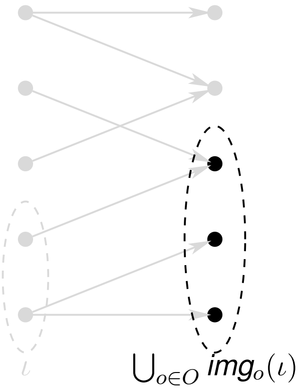
relational join of ι and $\mathcal{R}_1$



Symbolic reachability computation

$$\exists A^0.(\iota^0 \wedge \mathcal{R}_1(A^0, A^1))$$

projection to time 1: $\bigcup_{o \in O} \mathit{img}_o(\iota)$
This is the set of states that are reachable from ι by one step with o.
$\mathit{img}_o(\iota) = \{s' | sos', \iota \models s\}$



Symbolic reachability computation

$$\exists A^1.(\exists A^0.(\iota^0 \wedge \mathcal{R}_1(A^0, A^1)) \wedge \mathcal{R}_1(A^1, A^2))$$

States that are reachable by two steps:
$\bigcup_{o \in O} \mathit{img}_o(\bigcup_{o \in O} \mathit{img}_o(\iota))$

Symbolic reachability computation

$$\exists A^1.(\exists A^0.(\iota^0 \wedge \mathcal{R}_1(A^0, A^1)) \wedge \mathcal{R}_1(A^1, A^2)) \wedge G^2$$

Goal states that are reachable by two steps:
$(\bigcup_{o \in O} \textit{img}_o(\bigcup_{o \in O} \textit{img}_o(\iota))) \cap G$

Formula for plans of length 2:
$\iota^0 \wedge \mathcal{R}_1(A^0, A^1) \wedge \mathcal{R}_1(A^1, A^2) \wedge G^2$

Images by $\exists$-abstraction

Let

- $A = \{a_1, \ldots, a_n\}$,
- $A' = \{a'_1, \ldots, a'_n\}$,
- ϕ be a formula over A that represents a set T of states, and
- $\tau_A(o)$ the formula over $A \cup A'$ that represents the action o (a binary relation on states).

The image $\{s' \in S | s \in T, sos'\}$ of T with respect to o is represented by

$$\exists A.(\phi \wedge F(o))$$

which is a formula over A'.
To obtain a formula over A we rename the variables.

$$\mathit{img}_o(\phi) = (\exists A.(\phi \wedge F(o)))[A/A']$$

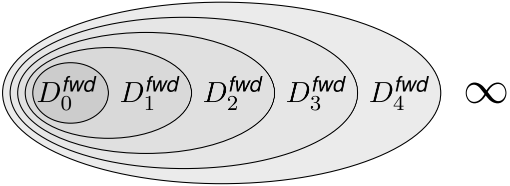
Images and preimages by formula manipulation

Definition

Let o be an action and ϕ a formula. Define

$$\mathit{img}_o(\phi) = (\exists A.(\phi \wedge F(o)))[A/A']$$
$$\mathit{preimg}_o(\phi) = \exists A'.(F(o) \wedge \phi[A'/A])$$

Forward distances with formulae



Forward distances with formulae

$$D_0^{fwd} = I$$
$$D_i^{fwd} = D_{i-1}^{fwd} \vee \bigvee_{o \in O} img_o(D_{i-1}^{fwd}) \text{ for all } i \geq 1$$

Constructing a plan given the distances

Let n be the minimum number such that $D_n^{\mathit{fwd}} \wedge G$ is satisfiable. This means that the shortest plan has length n. An action sequence from an initial state to G can be extracted as follows (starting from its last action.)

1. Set $G := G \wedge D_n^{\mathit{fwd}}$.
2. Choose any action e such that $\mathit{preimg}_e(G) \wedge D_{n-1}^{\mathit{fwd}}$ is satisfiable.
3. Set $G := \mathit{preimg}_e(G) \wedge D_{n-1}^{\mathit{fwd}}$ and $n := n - 1$.
4. If $n > 0$ go to 2.

SAT vs. symbolic reachability

The formula in the SAT case is

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

In the symbolic reachability computation, generate

$$\begin{array}{l}
\iota^0 \\
\iota^0 \wedge \mathcal{R}_1(A^0, A^1) \\
\iota^0 \wedge \mathcal{R}_1(A^0, A^1) \wedge \mathcal{R}_1(A^1, A^2) \\
\vdots \\
\iota^0 \wedge \mathcal{R}_1(A^0, A^1) \wedge \mathcal{R}_1(A^1, A^2) \wedge \cdots \wedge \mathcal{R}_1(A^{t-1}, A^t)
\end{array}$$

and from each formula abstract away all but the last time point, and then intersect the resulting set with G to test if goals can be reached.

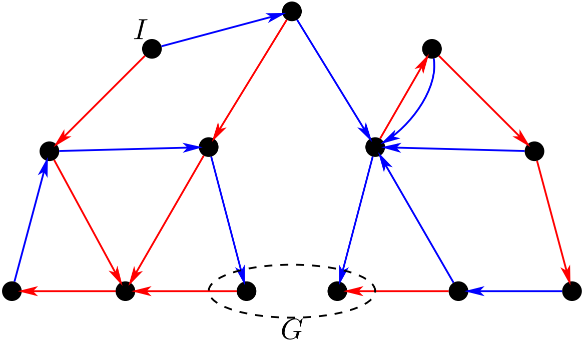
These are from the logical point of view exactly the same thing.

Planning by state-space search

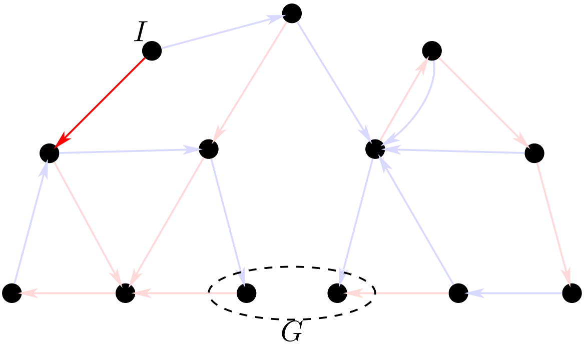
There are many alternative ways of doing planning by state-space search.

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

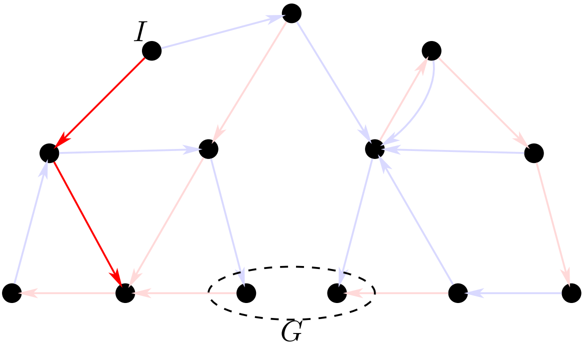
Planning by forward search
with depth-first search



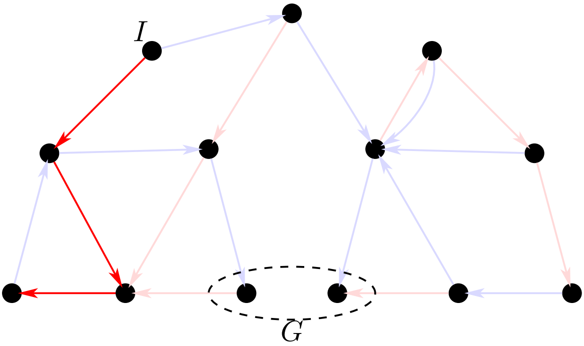
Planning by forward search
with depth-first search



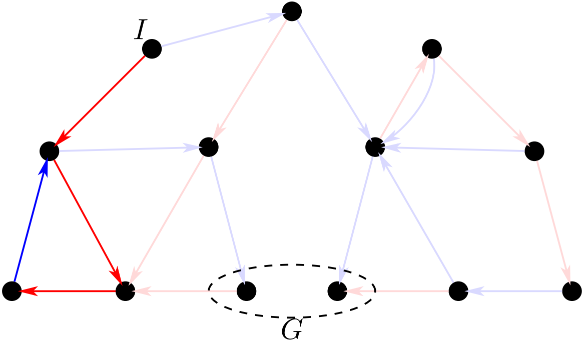
Planning by forward search
with depth-first search



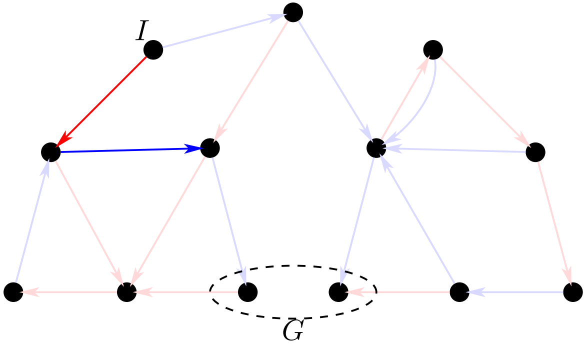
Planning by forward search
with depth-first search



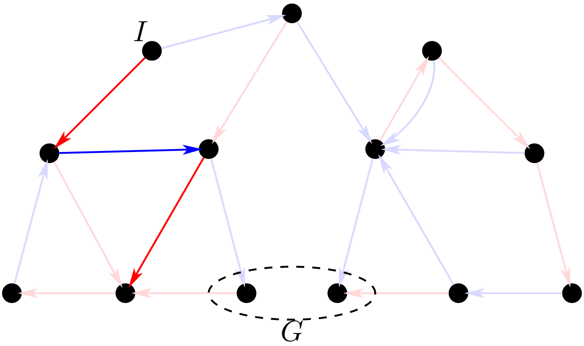
Planning by forward search
with depth-first search



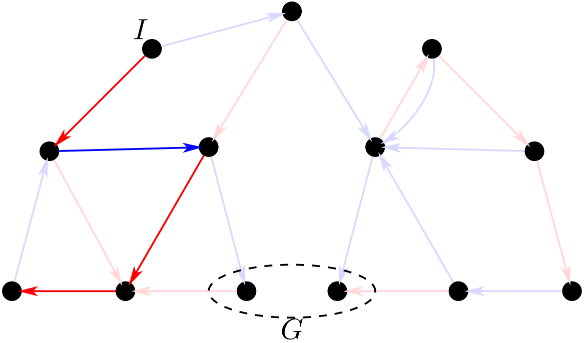
Planning by forward search
with depth-first search



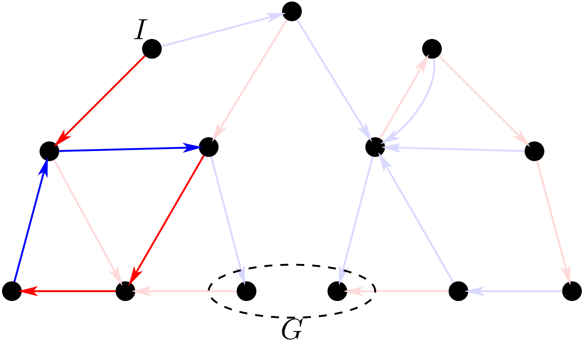
Planning by forward search
with depth-first search



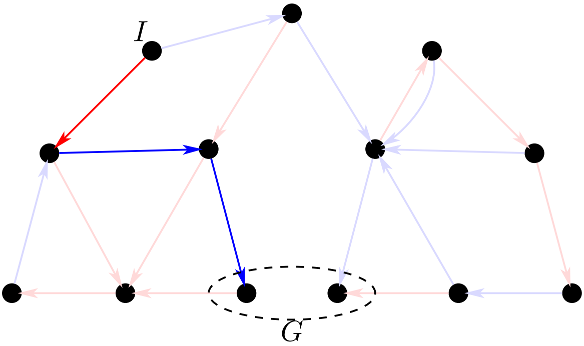
Planning by forward search
with depth-first search



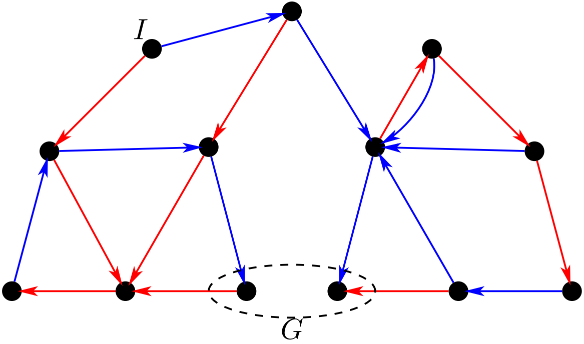
Planning by forward search
with depth-first search



Planning by forward search
with depth-first search

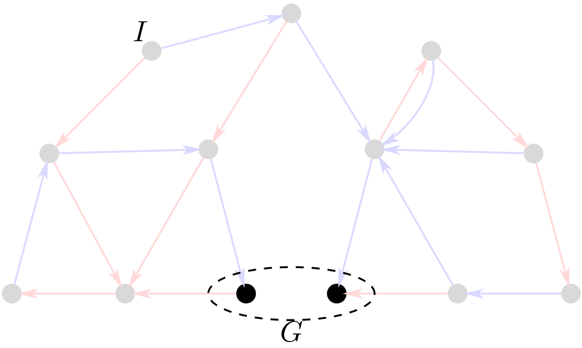


Planning by backward search
with depth-first search, for state sets



Planning by backward search
with depth-first search, for state sets

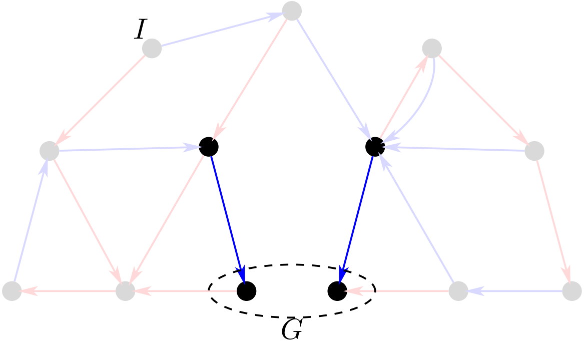
G



Planning by backward search
with depth-first search, for state sets

$G_1 = \mathit{regr}_{\longrightarrow}(G)$

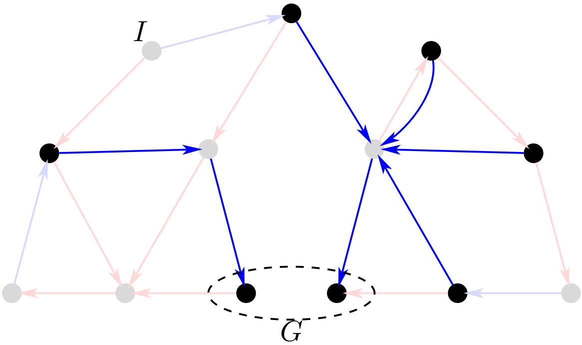
$G_1 \longrightarrow G$



Planning by backward search
with depth-first search, for state sets

$$G_1 = \textit{regr}_{\longrightarrow}(G)$$
$$G_2 = \textit{regr}_{\longrightarrow}(G_1)$$

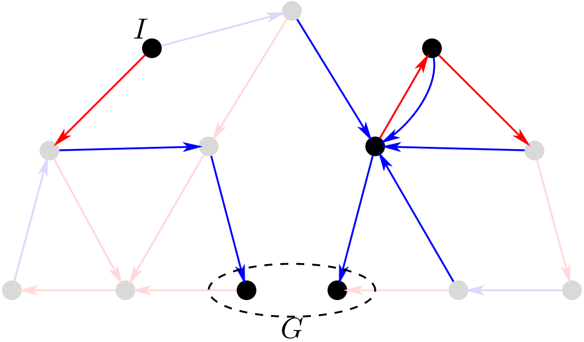
$$G_2 \dashrightarrow G_1 \dashrightarrow G$$



Planning by backward search
with depth-first search, for state sets

$$G_1 = \textit{regr}_{\textcolor{blue}{\longrightarrow}}(G)$$
$$G_2 = \textit{regr}_{\textcolor{blue}{\longrightarrow}}(G_1)$$
$$G_3 = \textit{regr}_{\textcolor{red}{\longrightarrow}}(G_2), I \models G_3$$

$$G_3 \textcolor{red}{\dashrightarrow} G_2 \textcolor{blue}{\dashrightarrow} G_1 \textcolor{blue}{\dashrightarrow} G$$



Progression

- Progression = computation of successor state(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.

Progression

For a given state s and action o with effects e, the successor state $exec_o(s)$ is obtained by changing the literals in $[e]_s$ true in s.

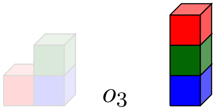
Regression

- Regression = computation of predecessors of states
- \+ Advantage over progression: a formula represents a set of states.
- \- More difficult to implement efficiently.

Regression

1. Start from ϕ which is initially set to G.
2. Repeat the following.
   1. First step: Choose an action o.
   2. Second step: Form a new goal $\phi := \mathit{preimg}_o(\phi)$.

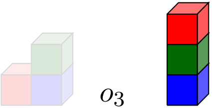
Regression for STRIPS actions
Example



$$o_3 = \langle \{ \textcolor{red}{\blacksquare}\text{onT}, \textcolor{red}{\blacksquare}\text{clr}, \textcolor{green}{\blacksquare}\text{clr} \}, \{ \neg \textcolor{green}{\blacksquare}\text{clr}, \neg \textcolor{red}{\blacksquare}\text{onT}, \textcolor{red}{\blacksquare}\text{on}\textcolor{green}{\blacksquare} \} \rangle$$

$$G = \{ \textcolor{red}{\blacksquare}\text{on}\textcolor{green}{\blacksquare}, \textcolor{green}{\blacksquare}\text{on}\textcolor{blue}{\blacksquare} \}$$

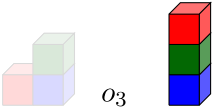
Regression for STRIPS actions
Example



$$o_3 = \langle\{\color{pink}\blacksquare\,\mathsf{onT}, \color{pink}\blacksquare\,\mathsf{clr}, \color{lightgreen}\blacksquare\,\mathsf{clr}\color{black}\}, \{\neg\color{lightgreen}\blacksquare\,\mathsf{clr}, \color{black}\neg\color{pink}\blacksquare\,\mathsf{onT}, \color{red}\blacksquare\,\color{black}\mathsf{on}\,\color{darkgreen}\blacksquare\color{black}\}\rangle$$

$$G = \{\color{red}\blacksquare\,\color{black}\mathsf{on}\,\color{darkgreen}\blacksquare\color{black}, \color{lightgreen}\blacksquare\,\mathsf{on}\,\color{lavender}\blacksquare\color{black}\}$$

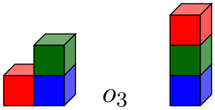
Regression for STRIPS actions
Example



$$o_3 = \langle \{ \blacksquare\text{onT}, \blacksquare\text{clr}, \blacksquare\text{clr} \}, \{ \neg\blacksquare\text{clr}, \neg\blacksquare\text{onT}, \blacksquare\text{on}\blacksquare \} \rangle$$

$$G = \{ \blacksquare\text{on}\blacksquare, \blacksquare\text{on}\blacksquare \}$$

Regression for STRIPS actions
Example

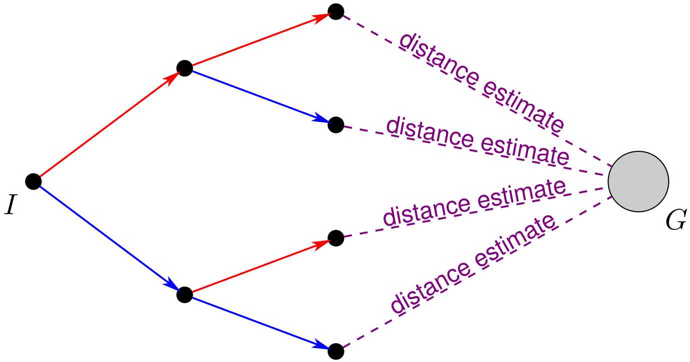


$$o_3 = \langle \{ {\color{red}\blacksquare}\text{onT}, {\color{red}\blacksquare}\text{clr}, {\color{green}\blacksquare}\text{clr} \}, \{ \neg {\color{green}\blacksquare}\text{clr}, \neg {\color{red}\blacksquare}\text{onT}, {\color{red}\blacksquare}\text{on}{\color{green}\blacksquare} \} \rangle$$

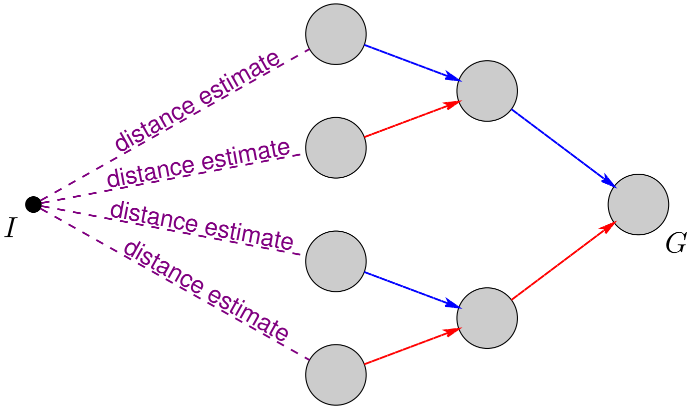
$$G = \{ {\color{red}\blacksquare}\text{on}{\color{green}\blacksquare}, {\color{green}\blacksquare}\text{on}{\color{blue}\blacksquare} \}$$

$$G_1 = \textit{regr}^{str}_{o_3}(G) = \{ {\color{green}\blacksquare}\text{on}{\color{blue}\blacksquare}, {\color{red}\blacksquare}\text{onT}, {\color{red}\blacksquare}\text{clr}, {\color{green}\blacksquare}\text{clr} \}$$

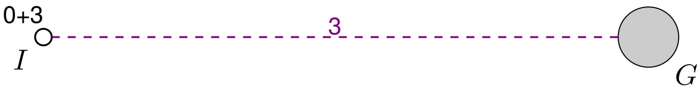
Planning by heuristic search
Forward search



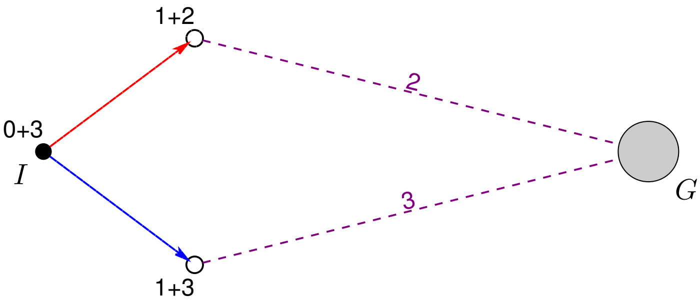
Planning by heuristic search
Backward search



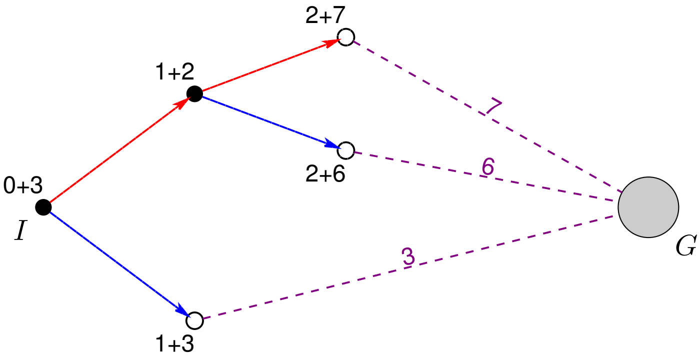
Search algorithms: A$*$
Example



Search algorithms: A∗
Example

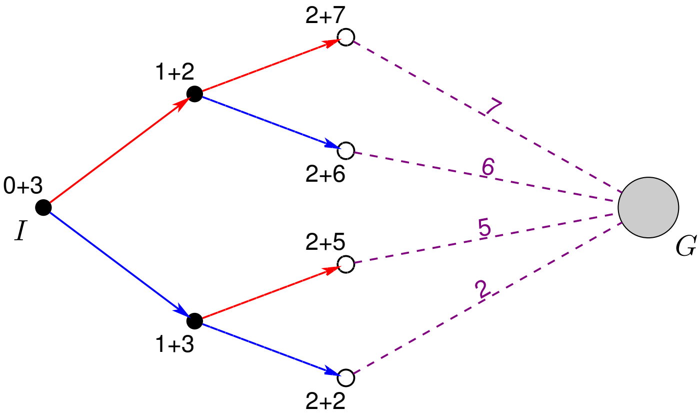


Search algorithms: A∗
Example

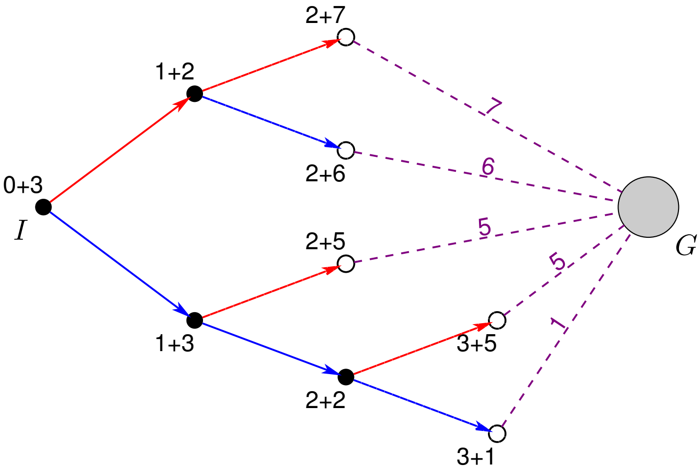


Search algorithms: A∗

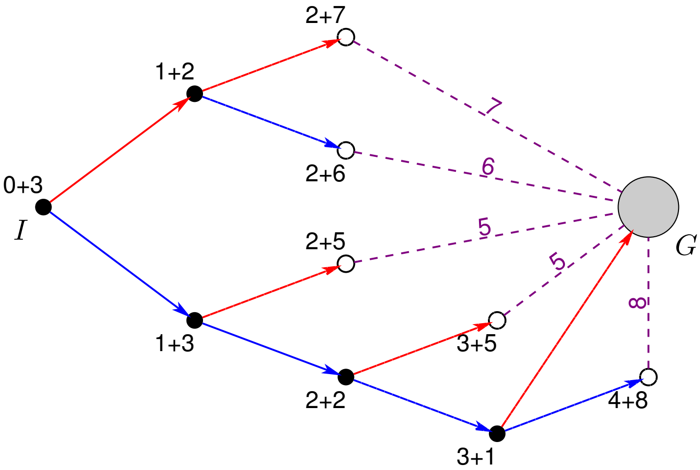
Example



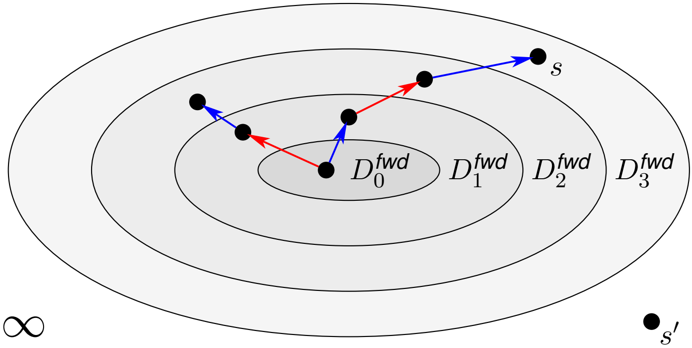
Search algorithms: A$*$
Example



Search algorithms: A∗
Example



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

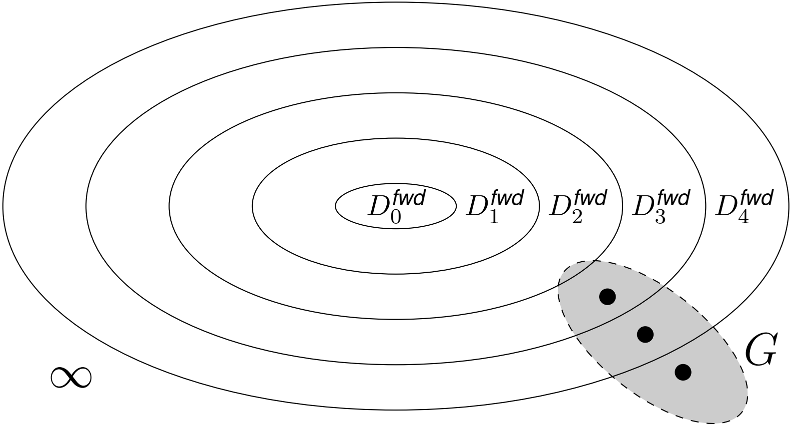


As $D_i^{\textit{fwd}} = D_3^{\textit{fwd}}$ for all $i > 3$, forward distance of state s' is ∞.

Introduction
Transition systems
Planning with SAT
Symbolic Methods
State-space search
Progression
Regression
Search
Heuristics

Distances
of formulas

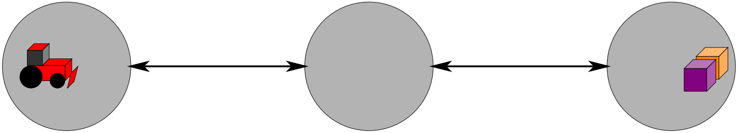
$\delta_I^{fwd}(G) = 3$ since $s \models G$ for some $s \in D_3^{fwd}$ and for no $s \in D_2^{fwd}$.



Distance estimation

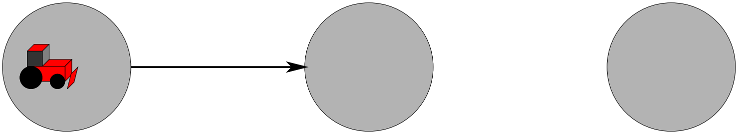
- Computation of exact distances is as hard as planning itself: only their approximations are useful as heuristics.
- We discuss a distance heuristic for controlling heuristic search algorithms like A$*$, IDA$*$.
- The distance estimates are a lower bound for forward distances: since they don't overestimate 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: $T12 = \langle T1, \{T2, \neg T1\}\rangle$
   - from 2 to 1: $T21 = \langle T2, \{T1, \neg T2\}\rangle$
   - from 2 to 3: $T23 = \langle T2, \{T3, \neg T2\}\rangle$
   - from 3 to 2: $T32 = \langle T3, \{T2, \neg T3\}\rangle$
2. Tractor pushes A:
   - from 2 to 1: $A21 = \langle T2 \wedge A2, \{T1, A1, \neg T2, \neg A2\}\rangle$
   - from 3 to 2: $A32 = \langle T3 \wedge A3, \{T2, A2, \neg T3, \neg A3\}\rangle$
3. Tractor pushes B:
   - from 2 to 1: $B21 = \langle T2 \wedge B2, \{T1, B1, \neg T2, \neg B2\}\rangle$
   - from 3 to 2: $B32 = \langle T3 \wedge B3, \{T2, B2, \neg T3, \neg B3\}\rangle$

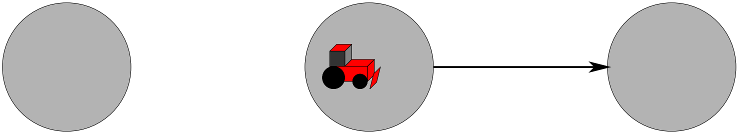
Distance estimation
Tractor example



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

Execute $T12 = \langle T1, \{T2, \neg T1\}\rangle$

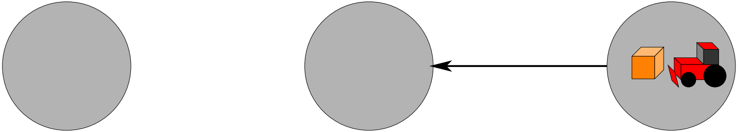
Distance estimation
Tractor example



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

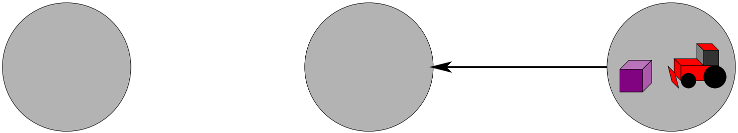
Execute $T23 = \langle T2, \{T3, \neg T2\}\rangle$

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

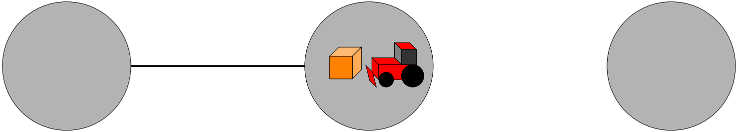
Execute $A32 = \langle T3 \wedge A3, \{T2, A2, \neg T3, \neg A3\}\rangle$



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

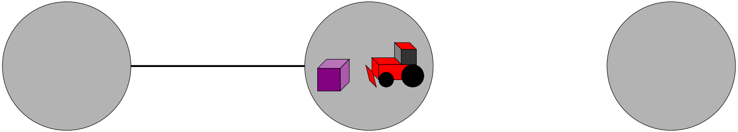
Execute $B32 = \langle T3 \wedge B3, \{T2, B2, \neg T3, \neg B3\}\rangle$

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

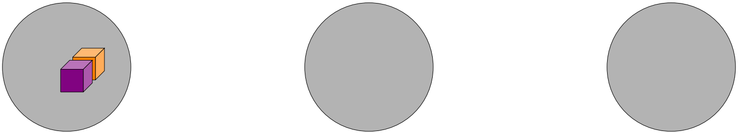
Execute $A21 = \langle T2 \wedge A2, \{T1, A1, \neg T2, \neg A2\}\rangle$



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

Execute $B21 = \langle T2 \wedge B2, \{T1, B1, \neg T2, \neg B2\}\rangle$

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.

Abstraction Heuristics

Key observation

Eliminating any state variable can only reduce the length of the shortest plan.

- Any abstraction, with some variables eliminated, yields a smaller state space.
- Distances in the abstract state space are lower bounds for the distances in the state space itself.

Abstraction Heuristics

The tractor example, abstracted to $\{A1, A2, A3, B1, B2, B3\}$ (eliminating the tractor) yields actions

1. Tractor moves:
   - from 1 to 2: $T12 = \langle \top, \{\} \rangle$
   - from 2 to 1: $T21 = \langle \top, \{\} \rangle$
   - from 2 to 3: $T23 = \langle \top, \{\} \rangle$
   - from 3 to 2: $T32 = \langle \top, \{\} \rangle$
2. Tractor pushes A:
   - from 2 to 1: $A21 = \langle A2, \{A1, \neg A2\} \rangle$
   - from 3 to 2: $A32 = \langle A3, \{A2, \neg A3\} \rangle$
3. Tractor pushes B:
   - from 2 to 1: $B21 = \langle B2, \{B1, \neg B2\} \rangle$
   - from 3 to 2: $B32 = \langle B3, \{B2, \neg B3\} \rangle$

The abstract state space has 9 states (as opposed to 27). Reaching $A1, B1$ from the abstract initial state $A3, B3$ takes 4 abstract actions.

Abstraction Heuristics
Aggregation of several abstractions

In practice it is only possible to use abstractions that retain only very few state variables. These typically yield very weak lower bounds.

Useful strategy: aggregate several abstractions.

1. Maximum of lower bounds from different abstractions
2. Sum of lower bounds from different abstractions, provided that no action gets counted twice.
3. More sophisticated aggregation methods exist.

Central problem: Which abstractions to aggregate?