Principles of AI Planning

Conformant planning

Malte Helmert Bernhard Nebel

Albert-Ludwigs-Universität Freiburg

January 31st, 2007

Reminder: Restrictions on observability

Let $\langle A, I, O, G, V \rangle$ be a problem instance in nondeterministic planning.

1. If $A = V$, the problem instance is fully observable.
2. If $V = \emptyset$, the problem instance is unobservable.
3. If there are no restrictions on V then the problem instance is partially observable.

Reminder: Restrictions on observability

Let $\langle A, I, O, G, V \rangle$ be a problem instance in nondeterministic planning.

1. If $A = V$, the problem instance is fully observable.
2. If $V = \emptyset$, the problem instance is unobservable.
3. If there are no restrictions on V then the problem instance is partially observable.

Reminder: Restrictions on observability

Let $\langle A, I, O, G, V \rangle$ be a problem instance in nondeterministic planning.

1. If $A = V$, the problem instance is fully observable.
2. If $V = \emptyset$, the problem instance is unobservable.
3. If there are no restrictions on V then the problem instance is partially observable.

Planning without observability: conformant planning

- Here we consider the second special case of planning with partial observability: planning without observability.
- Plans are sequences of actions because observations are not possible, actions cannot depend on the nondeterministic events or uncertain initial state, and hence the same actions have to be taken no matter what happens.
- Techniques needed for planning without observability can often be generalized to the general partially observable case.

Why acting without observation?

- Conformant planning is like planning to act in an environment while you are blind and deaf.
- Observations could be expensive or it is preferable to have a simple plan.
- Example: Finding synchronization sequences in hardware circuits
- Example: Initializing a system consisting of many components that are in unknown states.
- Internal motivation: try to understand the unobservable case so that one can better deal with the more complicated partially observable case.

Belief states and the belief space

- The current state is not in general known during plan execution. Instead, a set of possible current states is known.
- The set of possible current states forms the belief state.
- The set of all belief states is the belief space.
- If there are n states and none of them can be observationally distinguished from another, then there are $2^n - 1$ belief states.

Belief states and the belief space

- The current state is not in general known during plan execution. Instead, a set of possible current states is known.
- The set of possible current states forms the belief state.
- The set of all belief states is the belief space.
- If there are n states and none of them can be observationally distinguished from another, then there are $2^n - 1$ belief states.

Belief states and the belief space

- The current state is not in general known during plan execution. Instead, a set of possible current states is known.
- The set of possible current states forms the belief state.
- The set of all belief states is the belief space.
- If there are n states and none of them can be observationally distinguished from another, then there are $2^n - 1$ belief states.

Belief states and the belief space

- The current state is not in general known during plan execution. Instead, a set of possible current states is known.
- The set of possible current states forms the belief state.
- The set of all belief states is the belief space.
- If there are n states and none of them can be observationally distinguished from another, then there are $2^n - 1$ belief states.

The belief space

1. Let B be a belief state (a set of states).
2. Operator o is executable in B if it is executable in every $s \in B$.
3. When o is executed, possible next states are $T = img_o(B)$.
4. Belief states can be succinctly represented using Boolean formulae or BDDs.

The belief space

1. Let B be a belief state (a set of states).
2. Operator o is executable in B if it is executable in every $s \in B$.
3. When o is executed, possible next states are $T = img_o(B)$.
4. Belief states can be succinctly represented using Boolean formulae or BDDs.

The belief space

1. Let B be a belief state (a set of states).
2. Operator o is executable in B if it is executable in every $s \in B$.
3. When o is executed, possible next states are $T = \mathit{img}_o(B)$.
4. Belief states can be succinctly represented using Boolean formulae or BDDs.

The belief space

1. Let B be a belief state (a set of states).
2. Operator o is executable in B if it is executable in every $s \in B$.
3. When o is executed, possible next states are $T = \mathit{img}_o(B)$.
4. Belief states can be succinctly represented using Boolean formulae or BDDs.

The belief space
Example

Example (Next slide)

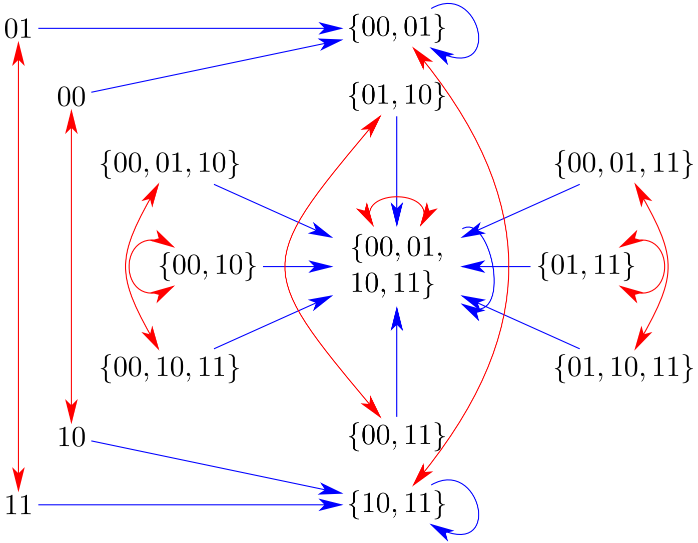
Belief space generated by states over two Boolean state variables.

$n = 2$ state variables, $2^n = 4$ states, $2^{2^n} - 1 = 15$ belief states

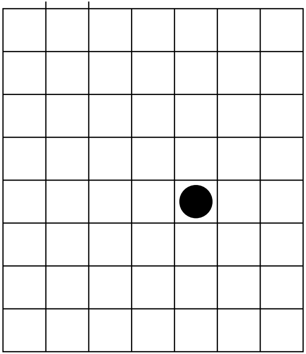
red action: complement the value of the first state variable

blue action: assign a random value to the second state variable

The belief space
Example

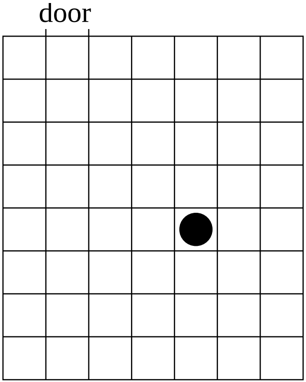


- A robot without any sensors, anywhere in a room of size 7×8.
- Actions: go North, South, East, West; if no way, just stay where you are
- Plan for getting out: $6 \times$ West, $7 \times$ North, $1 \times$ East, $1 \times$ North
- On the next slides we depict one possible location of the robot ($\bullet$) and the change in the belief state at every execution step by gray fields.



The belief space
Example

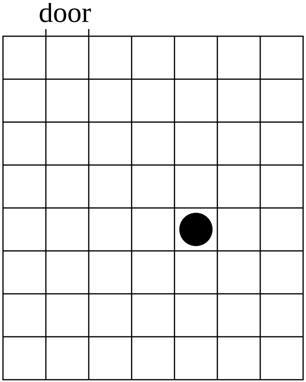
- A robot without any sensors, anywhere in a room of size 7×8.
- Actions: go North, South, East, West; if no way, just stay where you are
- Plan for getting out: $6 \times$ West, $7 \times$ North, $1 \times$ East, $1 \times$ North
- On the next slides we depict one possible location of the robot (•) and the change in the belief state at every execution step by gray fields.



The belief space

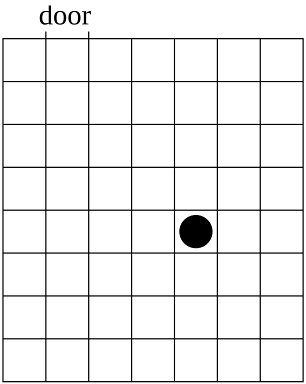
Example

- A robot without any sensors, anywhere in a room of size 7×8.
- Actions: go North, South, East, West; if no way, just stay where you are
- Plan for getting out: $6 \times$ West, $7 \times$ North, $1 \times$ East, $1 \times$ North
- On the next slides we depict one possible location of the robot (•) and the change in the belief state at every execution step by gray fields.

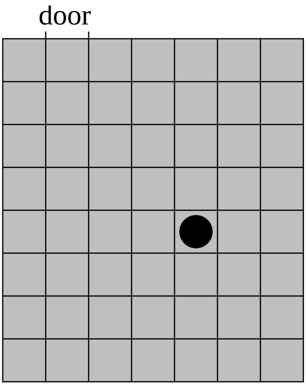


The belief space
Example

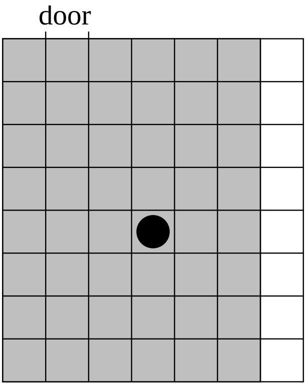
- A robot without any sensors, anywhere in a room of size 7×8.
- Actions: go North, South, East, West; if no way, just stay where you are
- Plan for getting out: $6 \times$ West, $7 \times$ North, $1 \times$ East, $1 \times$ North
- On the next slides we depict one possible location of the robot ($\bullet$) and the change in the belief state at every execution step by gray fields.



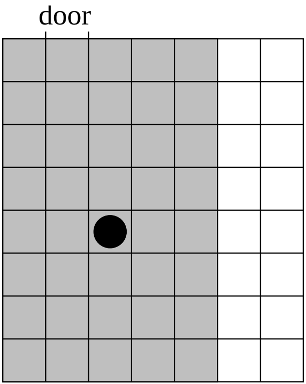
Example: belief state initially



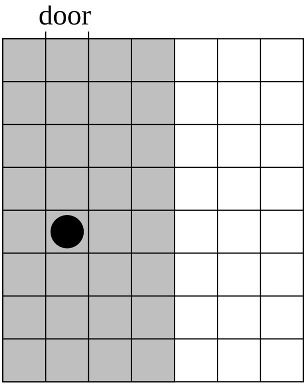
Example: belief state after W



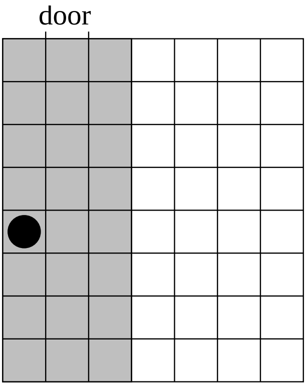
Example: after WW



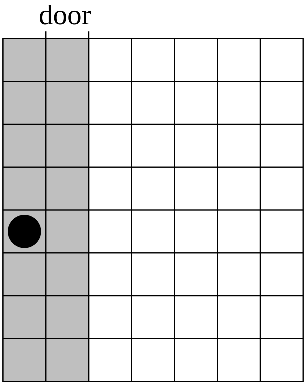
Example: after WWW



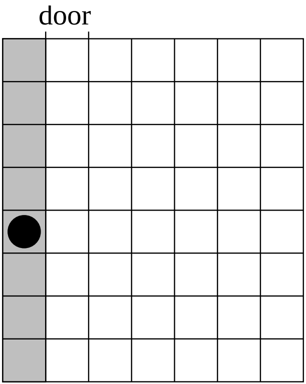
Example: after WWWW



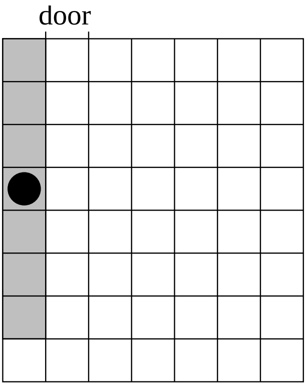
Example: after WWWWW



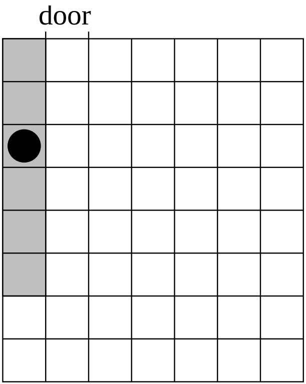
Example: after WWWWWW



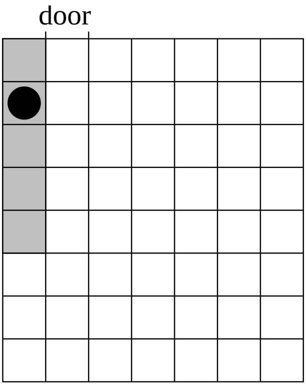
Example: after WWWWWWN



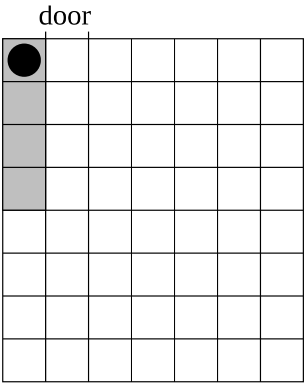
Example: after WWWWWWWNN



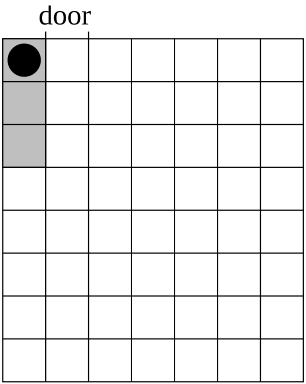
Example: after WWWWWWNNN



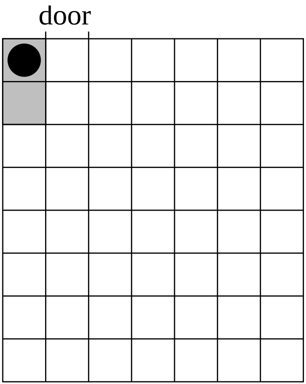
Example: after WWWWWWNNNN



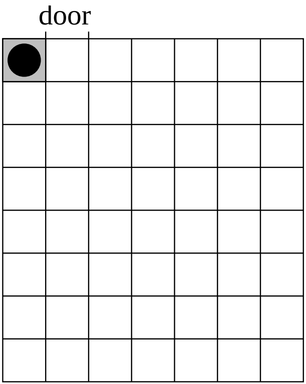
Example: after WWWWWWNNNNN



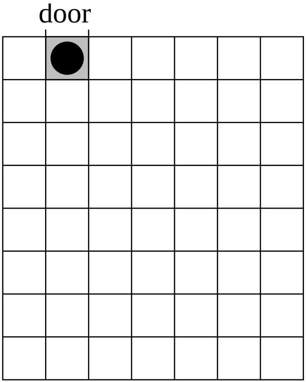
Example: after WWWWWWNNNNNN



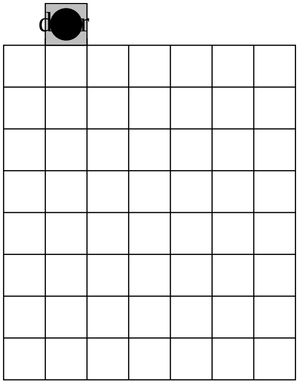
Example: after WWWWWWNNNNNNN



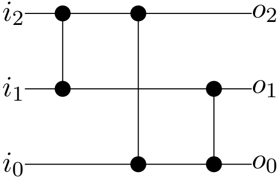
Example: after WWWWWWNNNNNNNE

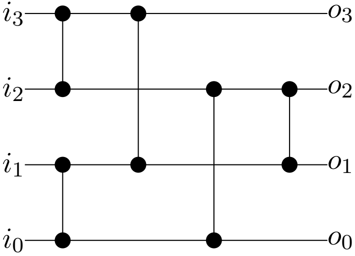


Example: after WWWWWWNNNNNNNEN



Sorting networks consist of comparator-swapper elements that compare the values of two inputs and output them sorted: if first input is bigger than the second, then they are swapped, otherwise the outputs are the inputs.
A sorting network for n inputs should sort any input sequence.





The belief space

Sorting networks

Theorem

If a sorting network correctly sorts any sequence of binary digits 0 and 1, then it correctly sorts any input sequence.

3-input sorting networks can be formalized as a succinct transition system $\langle A, I, O, G, V \rangle$ where

$$
\begin{aligned}
A &= \{a_0, a_1, a_2\} \\
I &= \top \\
O &= \{o_{01}, o_{02}, o_{12}\} \\
G &= (a_0 \rightarrow a_1) \wedge (a_1 \rightarrow a_2) \\
o_{01} &= \langle \top, (a_0 \wedge \neg a_1) \rhd (\neg a_0 \wedge a_1) \rangle \\
o_{02} &= \langle \top, (a_0 \wedge \neg a_2) \rhd (\neg a_0 \wedge a_2) \rangle \\
o_{12} &= \langle \top, (a_1 \wedge \neg a_2) \rhd (\neg a_1 \wedge a_2) \rangle
\end{aligned}
$$

The belief space
Sorting networks

A plan for the 3-input sorting network is o_{12}, o_{02}, o_{01}.
The initial states are $000, 001, 010, 011, 100, 101, 110, 111$.
The goal states are $000, 001, 011, 111$
The belief state evolves as follows.

$000, 001, 010, 011, 100, 101, 110, 111$	initially
$000, 001, 010, 011, 100, 101, 110, 111$	after o_{12}
$000, 001, 010, 011, 100, 101, 110, 111$	after o_{02}
$000, 001, 010, 011, 100, 101, 110, 111$	after o_{01}

The belief space
Sorting networks

A plan for the 3-input sorting network is o_{12}, o_{02}, o_{01}.
The initial states are $000, 001, 010, 011, 100, 101, 110, 111$.
The goal states are $000, 001, 011, 111$
The belief state evolves as follows.

$000, 001, 010, 011, 100, 101, 110, 111$	initially
$000, 001, 010, 011, 100, 101, 110, 111$	after o_{12}
$000, 001, 010, 011, 100, 101, 110, 111$	after o_{02}
$000, 001, 010, 011, 100, 101, 110, 111$	after o_{01}

A plan for the 3-input sorting network is o_{12}, o_{02}, o_{01}.
The initial states are $000, 001, 010, 011, 100, 101, 110, 111$.
The goal states are $000, 001, 011, 111$
The belief state evolves as follows.

$000, 001, 010, 011, 100, 101, 110, 111$	initially
$000, 001, 010, 011, 100, 101, 110, 111$	after o_{12}
$000, 001, 010, 011, 100, 101, 110, 111$	after o_{02}
$000, 001, 010, 011, 100, 101, 110, 111$	after o_{01}

The belief space
Sorting networks

A plan for the 3-input sorting network is o_{12}, o_{02}, o_{01}.
The initial states are $000, 001, 010, 011, 100, 101, 110, 111$.
The goal states are $000, 001, 011, 111$
The belief state evolves as follows.

$000, 001, 010, 011, 100, 101, 110, 111$	initially
$000, 001, 010, 011, 100, 101, 110, 111$	after o_{12}
$000, 001, 010, 011, 100, 101, 110, 111$	after o_{02}
$000, 001, 010, 011, 100, 101, 110, 111$	after o_{01}

Algorithms for unobservable problems

1. Find an operator sequence $o_1, \ldots, o_n$ that reaches a state satisfying G starting from any state satisfying I.
2. o_1 must be applicable in all states $B_0 = \{s \in S | s \models I\}$ satisfying I.
   o_2 must be applicable in all states in $B_1 = img_{o_1}(B_0)$.
   o_i must be applicable in all states in $B_i = img_{o_i}(B_{i-1})$ for all $i \in \{1, \ldots, n\}$.
   Terminal states must be goal states:
   $B_n \subseteq \{s \in S | s \models G\}$.

Algorithms for unobservable problems

1. Find an operator sequence $o_1, \ldots, o_n$ that reaches a state satisfying G starting from any state satisfying I.
2. o_1 must be applicable in all states $B_0 = \{s \in S | s \models I\}$ satisfying I.
   o_2 must be applicable in all states in $B_1 = img_{o_1}(B_0)$.
   o_i must be applicable in all states in $B_i = img_{o_i}(B_{i-1})$ for all $i \in \{1, \ldots, n\}$.
   Terminal states must be goal states:
   $B_n \subseteq \{s \in S | s \models G\}$.

Algorithms for unobservable problems

- Algorithms for deterministic planning can be lifted to the level of belief states.
- We can do forward search in the belief space with $img_o(B)$, backward search with $spreimg_o(B)$.
- We have already introduced implementation techniques that allow representing belief states B as formulae ϕ and computing images and pre-images respectively as $img_o(\phi)$ and $spreimg_o(\phi)$.
- Size of belief space is exponentially bigger than the size of the corresponding state space.
  For n state variables there are 2^n world states, and the belief space has a size of $2^{2^n} - 1$.
- Either explicit representation of world states or symbolic representation of a belief state using a BDD.

Algorithms for unobservable problems

- Algorithms for deterministic planning can be lifted to the level of belief states.
- We can do forward search in the belief space with $img_o(B)$, backward search with $spreimg_o(B)$.
- We have already introduced implementation techniques that allow representing belief states B as formulae ϕ and computing images and pre-images respectively as $img_o(\phi)$ and $spreimg_o(\phi)$.
- Size of belief space is exponentially bigger than the size of the corresponding state space.
  For n state variables there are 2^n world states, and the belief space has a size of $2^{2^n} - 1$.
- Either explicit representation of world states or symbolic representation of a belief state using a BDD.

Algorithms for unobservable problems

- Algorithms for deterministic planning can be lifted to the level of belief states.
- We can do forward search in the belief space with $\mathit{img}_o(B)$, backward search with $\mathit{spreimg}_o(B)$.
- We have already introduced implementation techniques that allow representing belief states B as formulae ϕ and computing images and pre-images respectively as $\mathit{img}_o(\phi)$ and $\mathit{spreimg}_o(\phi)$.
- Size of belief space is exponentially bigger than the size of the corresponding state space.
  For n state variables there are 2^n world states, and the belief space has a size of $2^{2^n} - 1$.
- Either explicit representation of world states or symbolic representation of a belief state using a BDD.

Algorithms for unobservable problems

- Algorithms for deterministic planning can be lifted to the level of belief states.
- We can do forward search in the belief space with *img*$_o(B)$, backward search with *spreimg*$_o(B)$.
- We have already introduced implementation techniques that allow representing belief states B as formulae ϕ and computing images and pre-images respectively as *img*$_o(\phi)$ and *spreimg*$_o(\phi)$.
- Size of belief space is exponentially bigger than the size of the corresponding state space.
  For n state variables there are 2^n world states, and the belief space has a size of $2^{2^n} - 1$.
- Either explicit representation of world states or symbolic representation of a belief state using a BDD.

Algorithms for unobservable problems

- Algorithms for deterministic planning can be lifted to the level of belief states.
- We can do forward search in the belief space with $\mathit{img}_o(B)$, backward search with $\mathit{spreimg}_o(B)$.
- We have already introduced implementation techniques that allow representing belief states B as formulae ϕ and computing images and pre-images respectively as $\mathit{img}_o(\phi)$ and $\mathit{spreimg}_o(\phi)$.
- Size of belief space is exponentially bigger than the size of the corresponding state space.
  For n state variables there are 2^n world states, and the belief space has a size of $2^{2^n} - 1$.
- Either explicit representation of world states or symbolic representation of a belief state using a BDD.

Heuristic search in belief space

progression/regression + heuristic search (A$*$, IDA$*$, simulated annealing, ...)

Heuristics:

- heuristic 1: backward distances (for forward search)
- heuristic 2: cardinality of belief state (for both forward and backward search)

Distance heuristics

Use backward distances of states as a heuristic:

$$\begin{aligned} D_0 &= G \\ D_{i+1} &= D_i \cup \textstyle\bigcup_{o \in O} \mathit{spreimg}_o(D_i) \text{ for all } i \geq 1 \end{aligned}$$

A lower bound on plan length for belief state B is j if $B \subseteq D_j$ and $B \not\subseteq D_{j-1}$ for $j \geq 1$.
This is an admissible heuristic (does not overestimate the distance).

Cardinality heuristics

- Backward search: Prefer operators that increase the size of the belief state, i.e. find a plan suffix that reaches a goal state from more starting states.
- Forward search: Prefer operators that decrease the size of the belief state, i.e. reduce the uncertainty about the current state and make reaching goals easier.
  For the room navigation example this heuristic works very well until the size of the belief state is 1.
- This heuristic is **not admissible**.
- Computing the cardinality of a belief state from its BDD representation takes linear time. (Propositional logic in general: problem is NP-hard.)
- Backward search with the cardinality heuristic seems to work particularly well on the examples from the literature.

Cardinality heuristics

- Backward search: Prefer operators that increase the size of the belief state, i.e. find a plan suffix that reaches a goal state from more starting states.
- Forward search: Prefer operators that decrease the size of the belief state, i.e. reduce the uncertainty about the current state and make reaching goals easier.
  For the room navigation example this heuristic works very well until the size of the belief state is 1.
- This heuristic is **not admissible**.
- Computing the cardinality of a belief state from its BDD representation takes linear time. (Propositional logic in general: problem is NP-hard.)
- Backward search with the cardinality heuristic seems to work particularly well on the examples from the literature.

Cardinality heuristics

- Backward search: Prefer operators that increase the size of the belief state, i.e. find a plan suffix that reaches a goal state from more starting states.
- Forward search: Prefer operators that decrease the size of the belief state, i.e. reduce the uncertainty about the current state and make reaching goals easier.
  For the room navigation example this heuristic works very well until the size of the belief state is 1.
- This heuristic is **not admissible**.
- Computing the cardinality of a belief state from its BDD representation takes linear time. (Propositional logic in general: problem is NP-hard.)
- Backward search with the cardinality heuristic seems to work particularly well on the examples from the literature.

Cardinality heuristics

- Backward search: Prefer operators that increase the size of the belief state, i.e. find a plan suffix that reaches a goal state from more starting states.
- Forward search: Prefer operators that decrease the size of the belief state, i.e. reduce the uncertainty about the current state and make reaching goals easier.
  For the room navigation example this heuristic works very well until the size of the belief state is 1.
- This heuristic is **not admissible**.
- Computing the cardinality of a belief state from its BDD representation takes linear time. (Propositional logic in general: problem is NP-hard.)
- Backward search with the cardinality heuristic seems to work particularly well on the examples from the literature.

Cardinality heuristics

- Backward search: Prefer operators that increase the size of the belief state, i.e. find a plan suffix that reaches a goal state from more starting states.
- Forward search: Prefer operators that decrease the size of the belief state, i.e. reduce the uncertainty about the current state and make reaching goals easier.
  For the room navigation example this heuristic works very well until the size of the belief state is 1.
- This heuristic is **not admissible**.
- Computing the cardinality of a belief state from its BDD representation takes linear time. (Propositional logic in general: problem is NP-hard.)
- Backward search with the cardinality heuristic seems to work particularly well on the examples from the literature.

Conformant-FF: Lazy representation of belief states

- Instead of computing the belief states (explicitly or symbolically), one could just store the representation of the initial state and the plan (as propositional formula) so far – a lazy representation
- Works particularly well when all *conditions* in STRIPS-form, i.e., conjunctions of atoms and deterministic operators (can be extended)
- Necessarily true atoms at each point in the plan can be computed using one UNSAT-call
- The FF heuristic h_{FF} can be extended to deal with belief state planning by using an unsound approximation of the propositional formula.

Conformant-FF: Lazy representation of belief states

- Instead of computing the belief states (explicitly or symbolically), one could just store the representation of the initial state and the plan (as propositional formula) so far – a lazy representation
- Works particularly well when all *conditions* in STRIPS-form, i.e., conjunctions of atoms and deterministic operators (can be extended)
- Necessarily true atoms at each point in the plan can be computed using one UNSAT-call
- The FF heuristic h_{FF} can be extended to deal with belief state planning by using an unsound approximation of the propositional formula.

Conformant-FF: Lazy representation of belief states

- Instead of computing the belief states (explicitly or symbolically), one could just store the representation of the initial state and the plan (as propositional formula) so far – a lazy representation
- Works particularly well when all *conditions* in STRIPS-form, i.e., conjunctions of atoms and deterministic operators (can be extended)
- Necessarily true atoms at each point in the plan can be computed using one UNSAT-call
- The FF heuristic h_{FF} can be extended to deal with belief state planning by using an unsound approximation of the propositional formula.

Conformant-FF: Lazy representation of belief states

- Instead of computing the belief states (explicitly or symbolically), one could just store the representation of the initial state and the plan (as propositional formula) so far – a lazy representation
- Works particularly well when all *conditions* in STRIPS-form, i.e., conjunctions of atoms and deterministic operators (can be extended)
- Necessarily true atoms at each point in the plan can be computed using one UNSAT-call
- The FF heuristic h_{FF} can be extended to deal with belief state planning by using an unsound approximation of the propositional formula.

Representing the plan as a CNF formula

- Let π be the plan $\langle o_1, \ldots, o_n \rangle$.
- The atoms a_i are indexed by the time point t, i.e., $a_i(t)$.
- The initial belief state (at time point 0) is presented as a CNF formula over literals indexed with 0.
- Given we have a representation for the plan up time point t, we extend the formula as follows:
  1. Effect axioms: For every effect e (in normal form) of o_{t+1} with $e = ((a_1 \wedge \ldots \wedge a_m) \rhd l)$, we insert the formula

     $$\neg a_1(t) \vee \ldots \vee \neg a_m(t) \vee l(t+1)$$

  2. Frame axioms: for every atom a, let $e_1, \ldots, e_n$ be the effects that contain a as a negative atomic effect; for every tuple $a_1, \ldots, a_n$ such that a_i is a part of e_i's effect condition, we insert

     $$\neg a(t) \vee a_1(t) \vee \ldots \vee a_n(t) \vee a(t+1).$$

     Similarly for positive atomic effects!

Representing the plan as a CNF formula

- Let π be the plan $\langle o_1, \dots, o_n \rangle$.
- The atoms a_i are indexed by the time point t, i.e., $a_i(t)$.
- The initial belief state (at time point 0) is presented as a CNF formula over literals indexed with 0.
- Given we have a representation for the plan up time point t, we extend the formula as follows:
  1. Effect axioms: For every effect e (in normal form) of o_{t+1} with $e = ((a_1 \wedge \dots \wedge a_m) \rhd l)$, we insert the formula
  $$\neg a_1(t) \vee \dots \vee \neg a_m(t) \vee l(t+1)$$
  2. Frame axioms: for every atom a, let $e_1, \dots, e_n$ be the effects that contain a as a negative atomic effect; for every tuple $a_1, \dots, a_n$ such that a_i is a part of e_i's effect condition, we insert
  $$\neg a(t) \vee a_1(t) \vee \dots \vee a_n(t) \vee a(t+1).$$
  Similarly for positive atomic effects!

Representing the plan as a CNF formula

- Let π be the plan $\langle o_1, \ldots, o_n \rangle$.
- The atoms a_i are indexed by the time point t, i.e., $a_i(t)$.
- The initial belief state (at time point 0) is presented as a CNF formula over literals indexed with 0.
- Given we have a representation for the plan up time point t, we extend the formula as follows:
  1. Effect axioms: For every effect e (in normal form) of o_{t+1} with $e = ((a_1 \wedge \ldots \wedge a_m) \rhd l)$, we insert the formula

     $$\neg a_1(t) \vee \ldots \vee \neg a_m(t) \vee l(t+1)$$

  2. Frame axioms: for every atom a, let $e_1, \ldots, e_n$ be the effects that contain a as a negative atomic effect; for every tuple $a_1, \ldots, a_n$ such that a_i is a part of e_i's effect condition, we insert

     $$\neg a(t) \vee a_1(t) \vee \ldots \vee a_n(t) \vee a(t+1).$$

     Similarly for positive atomic effects!

Representing the plan as a CNF formula

- Let π be the plan $\langle o_1, \ldots, o_n \rangle$.
- The atoms a_i are indexed by the time point t, i.e., $a_i(t)$.
- The initial belief state (at time point 0) is presented as a CNF formula over literals indexed with 0.
- Given we have a representation for the plan up time point t, we extend the formula as follows:
  1. Effect axioms: For every effect e (in normal form) of o_{t+1} with $e = ((a_1 \wedge \ldots \wedge a_m) \rhd l)$, we insert the formula
  $$\neg a_1(t) \vee \ldots \vee \neg a_m(t) \vee l(t+1)$$
  2. Frame axioms: for every atom a, let $e_1, \ldots, e_n$ be the effects that contain a as a negative atomic effect; for every tuple $a_1, \ldots, a_n$ such that a_i is a part of e_i's effect condition, we insert
  $$\neg a(t) \vee a_1(t) \vee \ldots \vee a_n(t) \vee a(t+1).$$
  Similarly for positive atomic effects!

Representing the plan as a CNF formula

- Let π be the plan $\langle o_1, \dots, o_n \rangle$.
- The atoms a_i are indexed by the time point t, i.e., $a_i(t)$.
- The initial belief state (at time point 0) is presented as a CNF formula over literals indexed with 0.
- Given we have a representation for the plan up time point t, we extend the formula as follows:
  1. Effect axioms: For every effect e (in normal form) of o_{t+1} with $e = ((a_1 \wedge \dots \wedge a_m) \rhd l)$, we insert the formula
$$\neg a_1(t) \vee \dots \vee \neg a_m(t) \vee l(t+1)$$
  2. Frame axioms: for every atom a, let $e_1, \dots, e_n$ be the effects that contain a as a negative atomic effect; for every tuple $a_1, \dots, a_n$ such that a_i is a part of e_i's effect condition, we insert
$$\neg a(t) \vee a_1(t) \vee \dots \vee a_n(t) \vee a(t+1).$$
     Similarly for positive atomic effects!

Representing the plan as a CNF formula

- Let π be the plan $\langle o_1, \ldots, o_n \rangle$.
- The atoms a_i are indexed by the time point t, i.e., $a_i(t)$.
- The initial belief state (at time point 0) is presented as a CNF formula over literals indexed with 0.
- Given we have a representation for the plan up time point t, we extend the formula as follows:
  1. Effect axioms: For every effect e (in normal form) of o_{t+1} with $e = ((a_1 \wedge \ldots \wedge a_m) \rhd l)$, we insert the formula
  $$\neg a_1(t) \vee \ldots \vee \neg a_m(t) \vee l(t+1)$$
  2. Frame axioms: for every atom a, let $e_1, \ldots, e_n$ be the effects that contain a as a negative atomic effect; for every tuple $a_1, \ldots, a_n$ such that a_i is a part of e_i's effect condition, we insert
  $$\neg a(t) \vee a_1(t) \vee \ldots \vee a_n(t) \vee a(t+1).$$
  Similarly for positive atomic effects!

Example

- Operator $o = \langle \top, (a \wedge c) \rhd \neg a \rangle$
- Initial belief state $= a \vee b, \neg a \vee \neg b, c$
- Plan $\pi = \langle o \rangle$
- Construction:
  1. Initial state: $a(0) \vee b(0), \neg a(0) \vee \neg b(0), c(0)$
  2. Effect axiom: $\neg a(0) \vee \neg c(0) \vee \neg a(1)$
  3. Simple positive frame axioms: $\neg b(0) \vee b(1), \neg c(0) \vee c(1)$
  4. Complex positive frame axioms: $\neg a(0) \vee a(0) \vee a(1), \neg a(0) \vee c(0) \vee a(1)$
  5. Simple negative frame axioms: $a(0) \vee \neg a(1), b(0) \vee \neg b(1), c(0) \vee \neg c(1)$

Example

- Operator $o = \langle \top, (a \wedge c) \rhd \neg a \rangle$
- Initial belief state $= a \vee b, \neg a \vee \neg b, c$
- Plan $\pi = \langle o \rangle$
- Construction:
  1. Initial state: $a(0) \vee b(0), \neg a(0) \vee \neg b(0), c(0)$
  2. Effect axiom: $\neg a(0) \vee \neg c(0) \vee \neg a(1)$
  3. Simple positive frame axioms: $\neg b(0) \vee b(1), \neg c(0) \vee c(1)$
  4. Complex positive frame axioms: $\neg a(0) \vee a(0) \vee a(1), \neg a(0) \vee c(0) \vee a(1)$
  5. Simple negative frame axioms: $a(0) \vee \neg a(1), b(0) \vee \neg b(1), c(0) \vee \neg c(1)$

Example

- Operator $o = \langle \top, (a \wedge c) \rhd \neg a \rangle$
- Initial belief state $= a \vee b, \neg a \vee \neg b, c$
- Plan $\pi = \langle o \rangle$
- Construction:
  1. Initial state: $a(0) \vee b(0), \neg a(0) \vee \neg b(0), c(0)$
  2. Effect axiom: $\neg a(0) \vee \neg c(0) \vee \neg a(1)$
  3. Simple positive frame axioms: $\neg b(0) \vee b(1), \neg c(0) \vee c(1)$
  4. Complex positive frame axioms: $\neg a(0) \vee a(0) \vee a(1), \neg a(0) \vee c(0) \vee a(1)$
  5. Simple negative frame axioms: $a(0) \vee \neg a(1), b(0) \vee \neg b(1), c(0) \vee \neg c(1)$

Example

- Operator $o = \langle \top, (a \wedge c) \rhd \neg a \rangle$
- Initial belief state $= a \vee b, \neg a \vee \neg b, c$
- Plan $\pi = \langle o \rangle$
- Construction:
  1. Initial state: $a(0) \vee b(0), \neg a(0) \vee \neg b(0), c(0)$
  2. Effect axiom: $\neg a(0) \vee \neg c(0) \vee \neg a(1)$
  3. Simple positive frame axioms: $\neg b(0) \vee b(1), \neg c(0) \vee c(1)$
  4. Complex positive frame axioms: $\neg a(0) \vee a(0) \vee a(1), \neg a(0) \vee c(0) \vee a(1)$
  5. Simple negative frame axioms: $a(0) \vee \neg a(1), b(0) \vee \neg b(1), c(0) \vee \neg c(1)$

Example

- Operator $o = \langle \top, (a \wedge c) \rhd \neg a \rangle$
- Initial belief state $= a \vee b, \neg a \vee \neg b, c$
- Plan $\pi = \langle o \rangle$
- Construction:
  1. Initial state: $a(0) \vee b(0), \neg a(0) \vee \neg b(0), c(0)$
  2. Effect axiom: $\neg a(0) \vee \neg c(0) \vee \neg a(1)$
  3. Simple positive frame axioms: $\neg b(0) \vee b(1), \neg c(0) \vee c(1)$
  4. Complex positive frame axioms: $\neg a(0) \vee a(0) \vee a(1), \neg a(0) \vee c(0) \vee a(1)$
  5. Simple negative frame axioms: $a(0) \vee \neg a(1), b(0) \vee \neg b(1), c(0) \vee \neg c(1)$

Example

- Operator $o = \langle \top, (a \wedge c) \rhd \neg a \rangle$
- Initial belief state $= a \vee b, \neg a \vee \neg b, c$
- Plan $\pi = \langle o \rangle$
- Construction:
  1. Initial state: $a(0) \vee b(0), \neg a(0) \vee \neg b(0), c(0)$
  2. Effect axiom: $\neg a(0) \vee \neg c(0) \vee \neg a(1)$
  3. Simple positive frame axioms: $\neg b(0) \vee b(1), \neg c(0) \vee c(1)$
  4. Complex positive frame axioms: $\neg a(0) \vee a(0) \vee a(1), \neg a(0) \vee c(0) \vee a(1)$
  5. Simple negative frame axioms: $a(0) \vee \neg a(1), b(0) \vee \neg b(1), c(0) \vee \neg c(1)$

Example

- Operator $o = \langle \top, (a \wedge c) \rhd \neg a \rangle$
- Initial belief state $= a \vee b, \neg a \vee \neg b, c$
- Plan $\pi = \langle o \rangle$
- Construction:
  1. Initial state: $a(0) \vee b(0), \neg a(0) \vee \neg b(0), c(0)$
  2. Effect axiom: $\neg a(0) \vee \neg c(0) \vee \neg a(1)$
  3. Simple positive frame axioms: $\neg b(0) \vee b(1), \neg c(0) \vee c(1)$
  4. Complex positive frame axioms: $\neg a(0) \vee a(0) \vee a(1), \neg a(0) \vee c(0) \vee a(1)$
  5. Simple negative frame axioms: $a(0) \vee \neg a(1), b(0) \vee \neg b(1), c(0) \vee \neg c(1)$

Example

- Operator $o = \langle \top, (a \wedge c) \rhd \neg a \rangle$
- Initial belief state $= a \vee b, \neg a \vee \neg b, c$
- Plan $\pi = \langle o \rangle$
- Construction:
  1. Initial state: $a(0) \vee b(0), \neg a(0) \vee \neg b(0), c(0)$
  2. Effect axiom: $\neg a(0) \vee \neg c(0) \vee \neg a(1)$
  3. Simple positive frame axioms: $\neg b(0) \vee b(1), \neg c(0) \vee c(1)$
  4. Complex positive frame axioms:
     $\neg a(0) \vee a(0) \vee a(1), \neg a(0) \vee c(0) \vee a(1)$
  5. Simple negative frame axioms:
     $a(0) \vee \neg a(1), b(0) \vee \neg b(1), c(0) \vee \neg c(1)$

Example

- Operator $o = \langle \top, (a \wedge c) \rhd \neg a \rangle$
- Initial belief state $= a \vee b, \neg a \vee \neg b, c$
- Plan $\pi = \langle o \rangle$
- Construction:
  1. Initial state: $a(0) \vee b(0), \neg a(0) \vee \neg b(0), c(0)$
  2. Effect axiom: $\neg a(0) \vee \neg c(0) \vee \neg a(1)$
  3. Simple positive frame axioms: $\neg b(0) \vee b(1), \neg c(0) \vee c(1)$
  4. Complex positive frame axioms: $\neg a(0) \vee a(0) \vee a(1), \neg a(0) \vee c(0) \vee a(1)$
  5. Simple negative frame axioms: $a(0) \vee \neg a(1), b(0) \vee \neg b(1), c(0) \vee \neg c(1)$

Computing the necessary true and false atoms

- In order to check whether an operator is applicable or the goal has been reached, one need to know, whether a set of atoms is necessarily true.
- Simply add $\neg a_i(t)$ and check for satisfiability. If it is unsatisfiable, a_i is necessarily true at time point t
- Necessarily true and false atoms can be cached to speed up reasoning.
- Problem: Designing a search heuristic in belief space!

Computing the necessary true and false atoms

- In order to check whether an operator is applicable or the goal has been reached, one need to know, whether a set of atoms is necessarily true.
- Simply add $\neg a_i(t)$ and check for satisfiability. If it is unsatisfiable, a_i is necessarily true at time point t
- Necessarily true and false atoms can be cached to speed up reasoning.
- Problem: Designing a search heuristic in belief space!

Computing the necessary true and false atoms

- In order to check whether an operator is applicable or the goal has been reached, one need to know, whether a set of atoms is necessarily true.
- Simply add $\neg a_i(t)$ and check for satisfiability. If it is unsatisfiable, a_i is necessarily true at time point t
- Necessarily true and false atoms can be cached to speed up reasoning.
- Problem: Designing a search heuristic in belief space!

Computing the necessary true and false atoms

- In order to check whether an operator is applicable or the goal has been reached, one need to know, whether a set of atoms is necessarily true.
- Simply add $\neg a_i(t)$ and check for satisfiability. If it is unsatisfiable, a_i is necessarily true at time point t
- Necessarily true and false atoms can be cached to speed up reasoning.
- Problem: Designing a search heuristic in belief space!

Extending the h_{FF} heuristics

- Reminder: FF computes the heuristic estimate by ignoring negative effects and trying to generate near-optimal plan for this relaxation.
- We do the same here and additionally …
- We over-approximate the clause set by reducing all clauses to two-literal clauses – randomly
- This theory is stronger, i.e. it is complete and most probably unsound
- Satisfiability can be solved in linear time

Extending the h_{FF} heuristics

- Reminder: FF computes the heuristic estimate by ignoring negative effects and trying to generate near-optimal plan for this relaxation.
- We do the same here and additionally ...
- We over-approximate the clause set by reducing all clauses to two-literal clauses – randomly
- This theory is stronger, i.e. it is complete and most probably unsound
- Satisfiability can be solved in linear time

Extending the h_{FF} heuristics

- Reminder: FF computes the heuristic estimate by ignoring negative effects and trying to generate near-optimal plan for this relaxation.
- We do the same here and additionally ...
- We over-approximate the clause set by reducing all clauses to two-literal clauses – randomly
- This theory is stronger, i.e. it is complete and most probably unsound
- Satisfiability can be solved in linear time

Extending the h_{FF} heuristics

- Reminder: FF computes the heuristic estimate by ignoring negative effects and trying to generate near-optimal plan for this relaxation.
- We do the same here and additionally ...
- We over-approximate the clause set by reducing all clauses to two-literal clauses – randomly
- This theory is stronger, i.e. it is complete and most probably unsound
- Satisfiability can be solved in linear time

Extending the h_{FF} heuristics

- Reminder: FF computes the heuristic estimate by ignoring negative effects and trying to generate near-optimal plan for this relaxation.
- We do the same here and additionally ...
- We over-approximate the clause set by reducing all clauses to two-literal clauses – randomly
- This theory is stronger, i.e. it is complete and most probably unsound
- Satisfiability can be solved in linear time

Summary

- Conformant planning is planning in a non-deterministic context without observation
- The search space is the belief space, the space of all belief sets.
- Techniques from classical planning can be lifted to belief space search
- BDDs are one possibility to implement this kind of search and model counting appears to be a reasonable heuristics
- Another possibility is lazy representation of plans as in Conformant-FF