Nondeterministic planning (June 6, 2005)

Execution graphs

- To formalize more complicated planning problems and more complicated forms of plans we define execution graphs of a transition system + plan.
- An execution graph describes the possible states of execution and the transitions between them.
- For memoryless plans the execution states and states of the transition system coincide because the execution mechanism is simple.
- For more complex forms of plans (defined in later lectures when discussing planning without full observability) the execution states also include information that encodes memory from earlier states of execution.

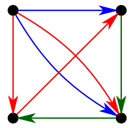
Execution graphs
Definition

Definition
Let $\langle S, I, O, G, P\rangle$ be a transition system with full observability and $\pi : S \to O$ a mapping from states to operators.
Then the execution graph is $\langle S, E\rangle$ where

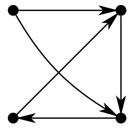
1. states $s \in S$ are the nodes of the graph,
2. $\langle s, s'\rangle \in E$ is an edge if and only if $s' \in img_{\pi(s)}(s)$,
3. the states $s \in I$ are the initial nodes,
4. the states $s \in G$ are the goal nodes,
5. nodes $s \in S$ such that $\langle s, s'\rangle \in E$ for no $s' \in S$ are terminal nodes.

Execution graphs
Example

Transition system



Execution graph with only infinite executions.



Plan objectives
Bounded reachability

- The simplest objective for nondeterministic planning is the one we have used in last lectures: reach a goal state with certainty.
- With this objective the nondeterminism can also be understood as an opponent like in 2-player games or in n-player games in general.
  Plan guarantees reaching a goal state no matter what the opponent does: plans are winning strategies.

Plan objectives
Bounded reachability

Definition
Let $\langle S, I, O, G, P\rangle$ be a transition system with full observability and $\pi : S \to O$ a mapping from states to operators.
Then π is a plan for bounded reachability if

> *all maximal paths starting from an initial node have a finite length and end in a goal node.*

This rules out infinite paths.

Need for unbounded executions / looping

- The first planning algorithm finds plans that reach a goal state without visiting any state twice.
- This property guarantees that the length of executions is bounded by some constant (which is smaller than the number of states.)
- Some solvable problems are not solvable this way.
  1. Action may fail to have any effect.
     Hit a coconut to break it.
  2. Action may fail and take us away from the goals.
     Build a house of cards.

  Consequences:
  1. It is impossible to avoid visiting some states several times.
  2. There is no finite upper bound on execution length.

Need for plans with unbounded executions

Assumption
1. For any nondeterministic effect $e_1|\cdots|e_n$ the probability of every effect $e_1, \ldots, e_n$ is > 0.
2. For any $s' \in img_o(s)$ the probability of reaching s' from s by o is > 0.

This assumption guarantees that any path in the execution graph has a non-zero probability.
This is not compatible with viewing nondeterminism as an opponent in a 2-player game: the opponent's strategy might rule out some of the choices $e_1, \ldots, e_n$.

Plan objectives
Unbounded reachability

Definition
Let $\langle S, I, O, G, P\rangle$ be a transition system with full observability and $\pi : S \to O$ a mapping from states to operators.
Then π is a plan for unbounded reachability if

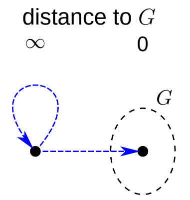
from every node to which there is a path from an initial node there is a path to a goal node that is a terminal node.

Looping
These plans may loop i.e. visit and revisit a state an unbounded number of times.
These plans even allow infinite executions that do not reach a goal state but the probability of such executions under the assumption we made is 0.

Need for plans with unbounded executions
Example

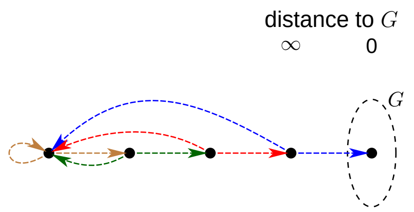
Example (Breaking a coconut)
- Initial state: coconut is intact.
- Goal state: coconut is broken.
- On every hit the coconut may or may not break.
- There is no finite upper bound on the number of hits.

This is equivalent to coin tossing.



Need for plans with unbounded executions
Example

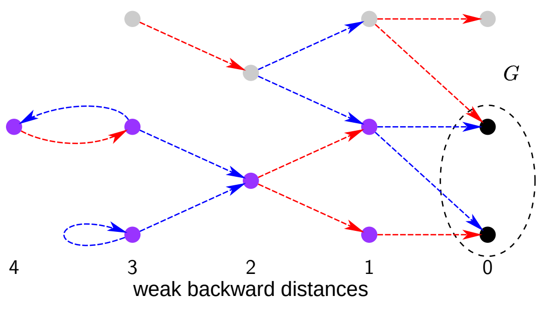
Example (Build a house of cards)
- Initial state: all cards lie on the table.
- Goal state: house of cards is complete.
- At every construction step the house may collapse.



Algorithm for unbounded reachability

- We give an algorithm that finds plans that may loop (unbounded reachability.)
- The algorithm is rather tricky in comparison to the algorithm for bounded reachability.
- Every state covered by a plan satisfies two properties:
  1. The state is good: there is at least one execution (= path in the execution graph) leading to a goal state.
  2. Every successor state is either a goal state or good.
- The algorithm repeatedly eliminates states that are not good.

Algorithm for unbounded reachability
Example



Subprocedure *prune*

- The procedure prune finds a maximal set of states for which reaching goals with looping is possible.
- Two nested loops.
  1. Inner loop identifies sets S_j of states from which a goal state can be reached with j steps without leaving the current set of candidate good states W_i.
     Limit of $S_0, S_1, \ldots$ will be W_{i+1}.
  2. Outer loop iterates through $i = 0, 1, 2, \ldots$ and produces a decreasing sequence of candidate good state sets $W_0, W_1, \ldots, W_n$ until $W_n = W_{n+1}$.

Subprocedure *prune*
Definition

PROCEDURE prune(T,O,G);
$W := T$;
REPEAT
  $W' := W$;
  $S := \emptyset$;
  REPEAT
    $S' := S$;
    $S := S' \cup \bigcup_{o \in O}$(*preimg*$_o(S' \cup G) \cap$ *spreimg*$_o(W' \cup G)$);
  UNTIL $S = S'$;
  $W := S$;
UNTIL $W = W'$;
RETURN W;

Subprocedure *prune*
Correctness

Lemma (Procedure prune)
Let S and $G \subseteq S$ be sets of states and O a set of operators Then prune(S,O,G) terminates after a finite number of steps and returns $W \subseteq S$ such that there is $\pi : W \to O$ such that

1. *for every $s \in W$ there is an execution $s_0, \ldots, s_n$ of π with $n \geq 1$ such that $s = s_0$ and $s_n \in G$,*
2. *img$_{\pi(s)}(\{s\}) \subseteq W \cup G$ for every $s \in W$, and*
3. *for every $s \in S \backslash W$ and function $\pi' : S \to O$ there is an execution $s_0, \ldots, s_n$ of π' such that $s = s_0$ and there is no $m \geq n$ and execution $s_n, \ldots, s_m$ such that $s_m \in G$.*

The planning algorithm

PROCEDURE FOplanLOOPS(I,O,G)
S := the set of all states;
L := $G \cup$ prune(S,O,G);
IF $I \not\subseteq L$ *THEN RETURN* $\emptyset$;
D_0 := G;
i := 1;
REPEAT *(* Compute weak backward distances *)*
  D_i := $D_{i-1} \cup \bigcup_{o \in O}$(*preimg*$_o(D_{i-1}) \cap$ *spreimg*$_o(L)$);
  $i := i + 1$;
UNTIL $D_i = D_{i-1}$;
FOR EACH $s \in D_i \backslash G$ *DO*
  d := number such that $s \in D_d \backslash D_{d-1}$;
  $\pi(s)$:= o such that *img*$_o(s) \subseteq L$ and *img*$_o(s) \cap D_{d-1} \neq \emptyset$;
END DO

Complexity of the planning algorithm

- The procedure *prune* runs in polynomial time in the number of states because the number of iterations of each loop is at most n – hence there are $O(n^2)$ iterations – and computation on each iteration takes polynomial time in the number of states.
- Finding conditional plans for full observability under the bounded and unbounded reachability objectives is in the complexity class EXPTIME.
- Lecture notes contain proofs showing that the planning problems are also EXPTIME-hard.

Maintenance goals

- Planning is often not about reaching a goal state in which execution can be terminated.
  1. An animal: find food, eat, sleep, find food, eat, sleep, ...
  2. Cleaner robot: keep the building clean.
- These problems cannot be directly formalized in terms of reachability because infinite (unbounded) plan execution is needed.
- We next formalize the simplest objective with infinite plan executions which is known as maintenance because the transition system has to be kept in the goal states indefinitely (the condition expressed by the goals has to be *maintained*.)

Plan objectives
Maintenance

Definition
Let $\langle S, I, O, G, P \rangle$ be a transition system with full observability and $\pi : S \to O$ a mapping from states to operators.
Then π is a plan for maintenance if

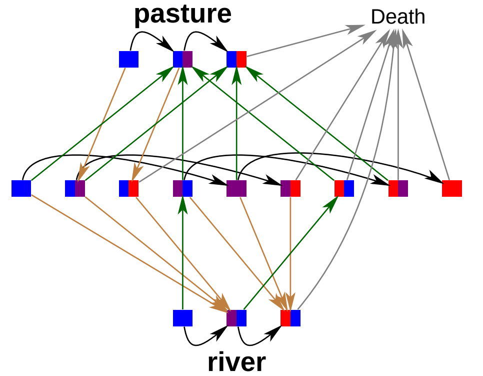
every node in the execution graph to which there is a path from an initial node is a goal node that has a successor node.

The execution graph does not have terminal nodes.

Maintenance
Example

- The state of an animal is determined by three state variables: hunger (0,1,2), thirst (0,1,2) and location (river, pasture, desert). There is also a special state called death.
- Thirst grows when not at river; at river it is 0.
- Hunger grows when not on pasture; on pasture it is 0.
- If hunger or thirst exceeds 2, the animal dies.
- The goal of the animal is to not die.

Algorithm for maintenance goals
Example



Maintenance goals
Example

We can infer rules backwards starting from the death condition.

1. If in desert and thirst = 2 must go to river.
2. If in desert and hunger = 2 must go to pasture.
3. If on pasture and thirst = 1 must go to desert.
4. If at river and hunger = 1 must go to desert.
5. ...

If the above rules conflict, the animal will die.
There is only one plan: go to pasture, go to desert, go to river, go to desert, ...

Algorithm for maintenance goals
Idea

1. Goal states are 0-safe: maintenance objective is satisfied for the current state.
2. Given all i-safe states, compute all $i + 1$-safe states: maintenance objective is satisfied for $i + 1$ time points.
3. $i + 1$-safe states can be computed from i-safe states by using strong preimages.
4. For some j, j-safe states equal $j + 1$-safe states because there are only finitely many states and at each step $j + 1$-safe states are a subset of j-safe states.
   Then j-safe states are also ∞-safe.

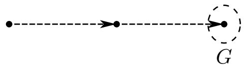
Summary of the algorithm idea
Repeatedly eliminate from consideration those states that in 1 or more steps unavoidably lead to a non-goal state.

Algorithm for maintenance goals
Definition

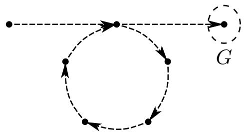
PROCEDURE FOplanMAINTENANCE(I,O,G)
G' := G;
REPEAT
  G'' := G';
  G' := $\bigcup_{o \in O}$ (*spreimg*$_o(G') \cap G$);
UNTIL $G' = G''$;
IF $I \not\subseteq G'$ *RETURN* $\emptyset$;
FOR EACH $s \in G'$ *DO*
  assign $\pi(s)$:= o such that *img*$_o(s) \subseteq G'$;
END DO
RETURN π;

Summary of objectives

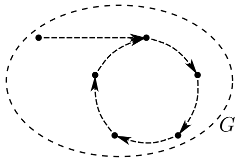
Bounded Reachability



Unbounded Reachability



Maintenance



Summary

- There are several possible objectives a plan can fulfill.
- We have considered bounded reachability, unbounded reachability and maintenance.
  The executions are respectively of bounded finite length, unbounded finite length, and infinite.
- These objectives have all been formalized in terms of the properties of execution graphs.
- We have presented dynamic-programming type (backward search) algorithms for all three planning problems.
- All three algorithms can be implemented by using binary decision diagrams BDDs as a data structure for state sets.