

# Principles of AI Planning

## 16. Strong cyclic planning

Albert-Ludwigs-Universität Freiburg



**UNI  
FREIBURG**

Bernhard Nebel and Robert Mattmüller

January 29th, 2014

# 1 Strong cyclic plans



- Motivation
- Nested Fixpoint Algorithm
- Incremental Planning Algorithm

Strong cyclic plans

Motivation

Nested Fixpoint Algorithm

Incremental Planning Algorithm

Maintenance

Summary

- The simplest objective for nondeterministic planning is the one we have considered in the previous lecture: reach a goal state with certainty.
- With this objective the nondeterminism can also be understood as **an opponent** like in 2-player games. The plan guarantees reaching a goal state no matter what the opponent does: plans are **winning strategies**.

Strong cyclic plans

Motivation

Nested Fixpoint Algorithm

Incremental Planning Algorithm

Maintenance

Summary

- In strong plans, goal states can be reached 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.

Strong cyclic plans

Motivation

Nested Fixpoint Algorithm

Incremental Planning Algorithm

Maintenance

Summary

### Fairness assumption

For any nondeterministic operator  $\langle \chi, \{e_1, \dots, e_n\} \rangle$ , the “probability” of every effect  $e_i$ ,  $i = 1, \dots, n$ , is greater than 0.

Alternatively: For each  $s' \in \text{img}_o(s)$  the “probability” of reaching  $s'$  from  $s$  by  $o$  is greater than 0.

This assumption guarantees that a strong cyclic plan reaches the goal **almost certainly** (with probability 1).

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  $e_1, \dots, e_n$ .

Strong cyclic plans

Motivation

Nested Fixpoint Algorithm

Incremental Planning Algorithm

Maintenance

Summary

# Need for strong cyclic plans

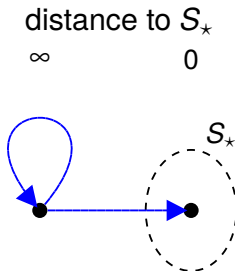
## Example



## 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.



Strong cyclic  
plans

Motivation

Nested Fixpoint  
Algorithm

Incremental  
Planning Algorithm

Maintenance

Summary

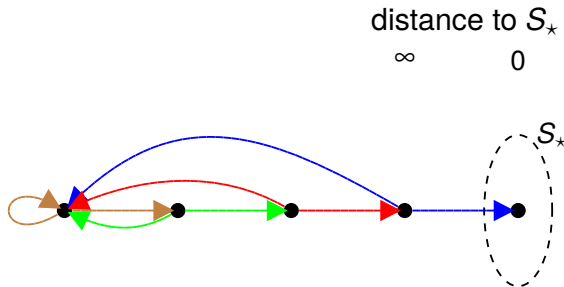
# Need for strong cyclic plans

## 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.



Strong cyclic  
plans

Motivation

Nested Fixpoint  
Algorithm

Incremental  
Planning Algorithm

Maintenance

Summary



We present two algorithms for strong cyclic planning:

- The **nested fixpoint algorithm** is conceptually simpler, but typically very costly, especially if not implemented symbolically.
  - Historically older
  - **Uninformed**
  - **Considers entire state space**
- The **determinization-based incremental planning algorithm** is a bit more complicated, but typically more efficient.
  - Historically newer, **state of the art**
  - Can use **informed** classical planner as sub-procedure
  - Often only **considers small portion of state space**

Strong cyclic plans

Motivation

Nested Fixpoint Algorithm

Incremental Planning Algorithm

Maintenance

Summary



- Finds plans that may loop (strong cyclic plans).
- The algorithm is rather tricky in comparison to the algorithm for strong plans.
- Every state covered by a plan satisfies two properties:
  - 1 The state is **good**: there is at least one execution (= path in the graph defined by the plan) 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.

Strong cyclic plans

Motivation

Nested Fixpoint Algorithm

Incremental Planning Algorithm

Maintenance

Summary

# Nested Fixpoint Algorithm

## Example



Strong cyclic  
plans

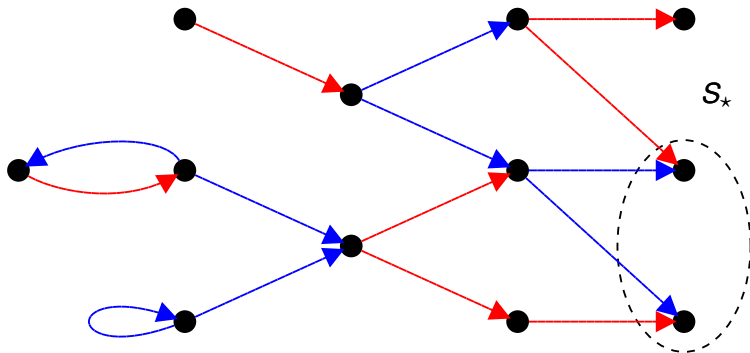
Motivation

**Nested Fixpoint  
Algorithm**

Incremental  
Planning Algorithm

Maintenance

Summary



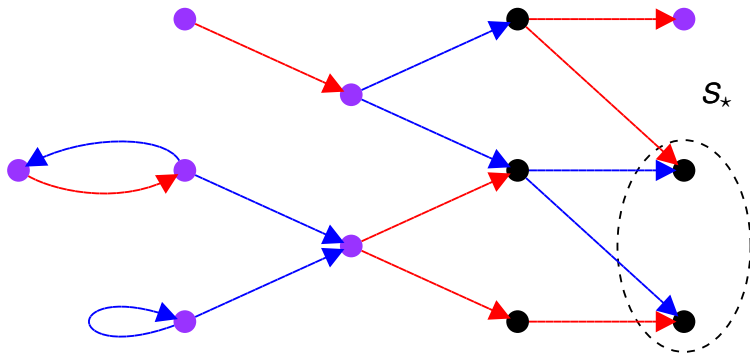


# Nested Fixpoint Algorithm

## Example



States from which goals are reachable in  $\leq 1$  steps so that all immediate successors are possibly good.



Strong cyclic  
plans

Motivation

Nested Fixpoint  
Algorithm

Incremental  
Planning Algorithm

Maintenance

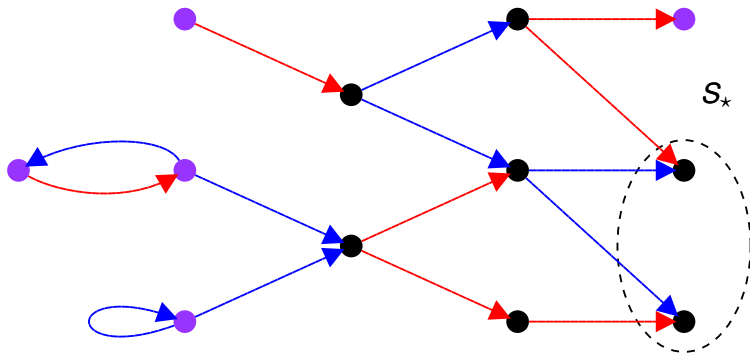
Summary

# Nested Fixpoint Algorithm

## Example



States from which goals are reachable in  $\leq 2$  steps so that all immediate successors are possibly good.



Strong cyclic  
plans

Motivation

**Nested Fixpoint  
Algorithm**

Incremental  
Planning Algorithm

Maintenance

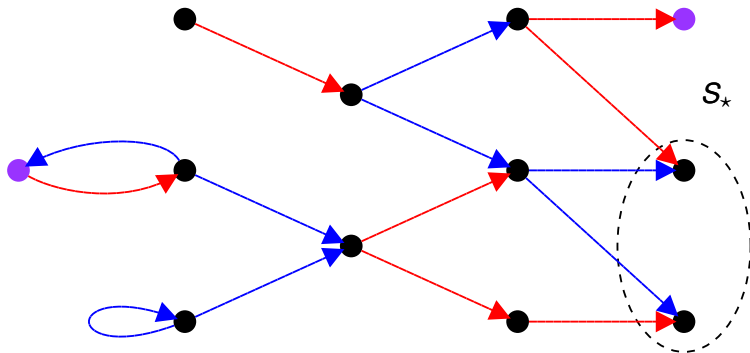
Summary

# Nested Fixpoint Algorithm

## Example



States from which goals are reachable in  $\leq 3$  steps so that all immediate successors are possibly good.



Strong cyclic  
plans

Motivation

Nested Fixpoint  
Algorithm

Incremental  
Planning Algorithm

Maintenance

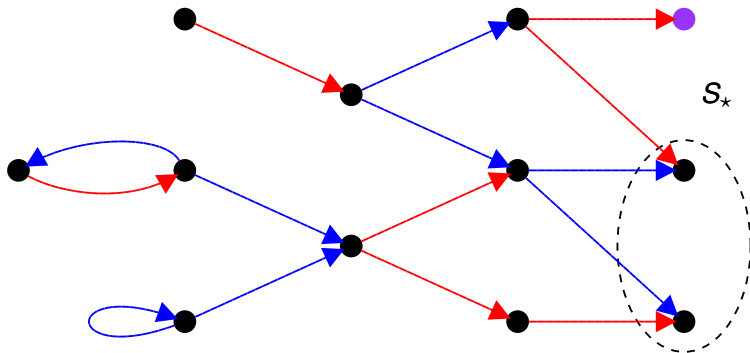
Summary

# Nested Fixpoint Algorithm

## Example



States from which goals are reachable in  $\leq 4$  steps so that all immediate successors are possibly good.



Strong cyclic plans

Motivation

Nested Fixpoint Algorithm

Incremental Planning Algorithm

Maintenance

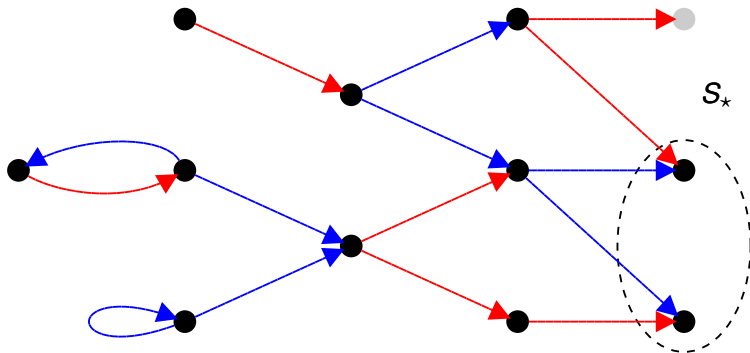
Summary

# Nested Fixpoint Algorithm

## Example



Eliminate states that turned out not to be good.



Strong cyclic  
plans

Motivation

**Nested Fixpoint  
Algorithm**

Incremental  
Planning Algorithm

Maintenance

Summary

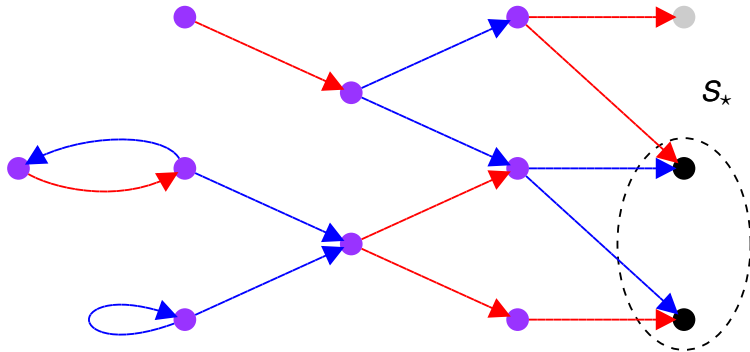


# Nested Fixpoint Algorithm

## Example



The set of possibly good states is now smaller.



Strong cyclic  
plans

Motivation

**Nested Fixpoint  
Algorithm**

Incremental  
Planning Algorithm

Maintenance

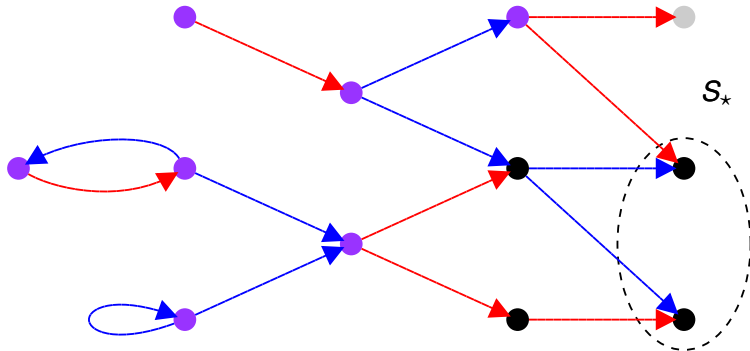
Summary

# Nested Fixpoint Algorithm

## Example



States from which goals are reachable in  $\leq 1$  steps so that all immediate successors are possibly good.



Strong cyclic  
plans

Motivation

**Nested Fixpoint  
Algorithm**

Incremental  
Planning Algorithm

Maintenance

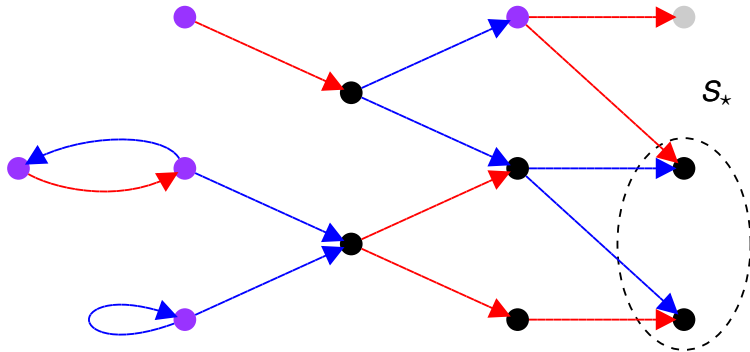
Summary

# Nested Fixpoint Algorithm

## Example



States from which goals are reachable in  $\leq 2$  steps so that all immediate successors are possibly good.



Strong cyclic  
plans

Motivation

Nested Fixpoint  
Algorithm

Incremental  
Planning Algorithm

Maintenance

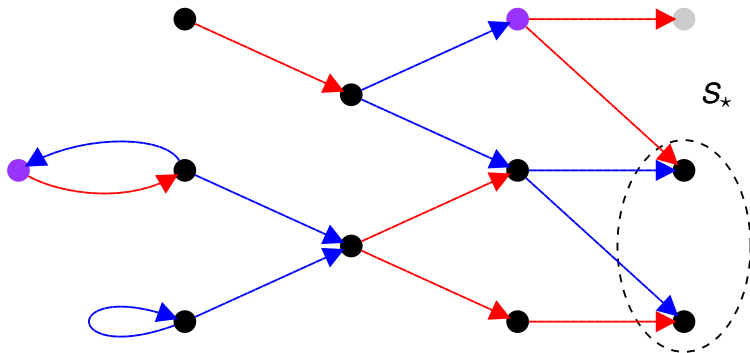
Summary

# Nested Fixpoint Algorithm

## Example



States from which goals are reachable in  $\leq 3$  steps so that all immediate successors are possibly good.



Strong cyclic  
plans

Motivation

Nested Fixpoint  
Algorithm

Incremental  
Planning Algorithm

Maintenance

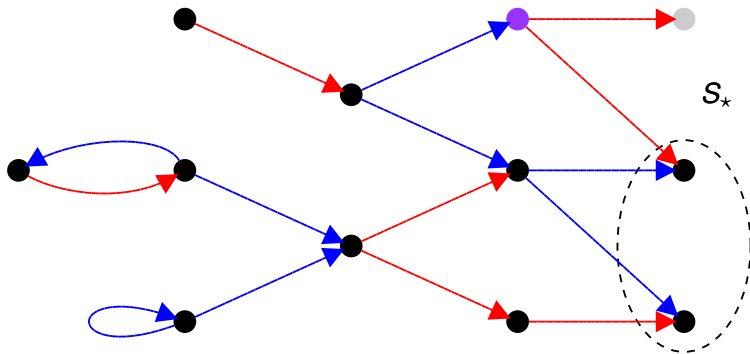
Summary

# Nested Fixpoint Algorithm

## Example



States from which goals are reachable in  $\leq 4$  steps so that all immediate successors are possibly good.



Strong cyclic  
plans

Motivation

Nested Fixpoint  
Algorithm

Incremental  
Planning Algorithm

Maintenance

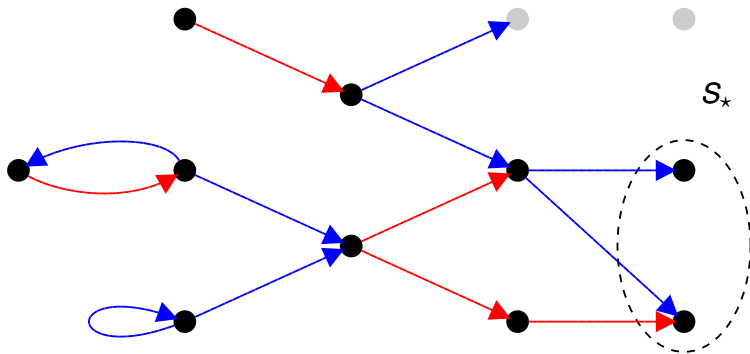
Summary

# Nested Fixpoint Algorithm

## Example



Eliminate states that turned out not to be good.



Strong cyclic plans

Motivation

**Nested Fixpoint Algorithm**

Incremental Planning Algorithm

Maintenance

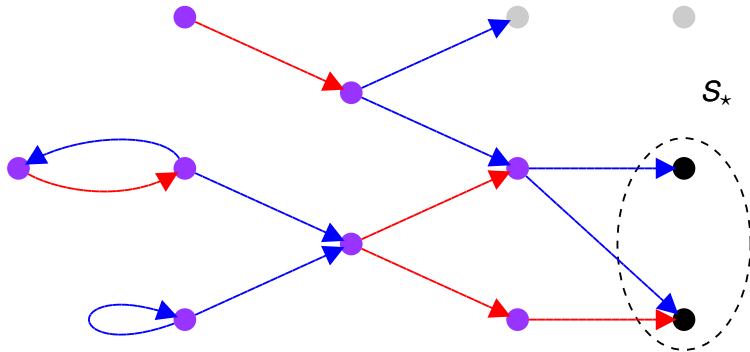
Summary

# Nested Fixpoint Algorithm

## Example



The set of possibly good states is now smaller.



Strong cyclic plans

Motivation

**Nested Fixpoint Algorithm**

Incremental Planning Algorithm

Maintenance

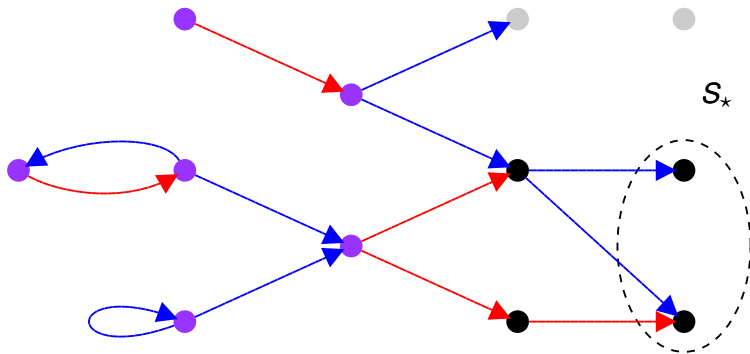
Summary

# Nested Fixpoint Algorithm

## Example



States from which goals are reachable in  $\leq 1$  steps so that all immediate successors are possibly good.



Strong cyclic  
plans

Motivation

**Nested Fixpoint  
Algorithm**

Incremental  
Planning Algorithm

Maintenance

Summary

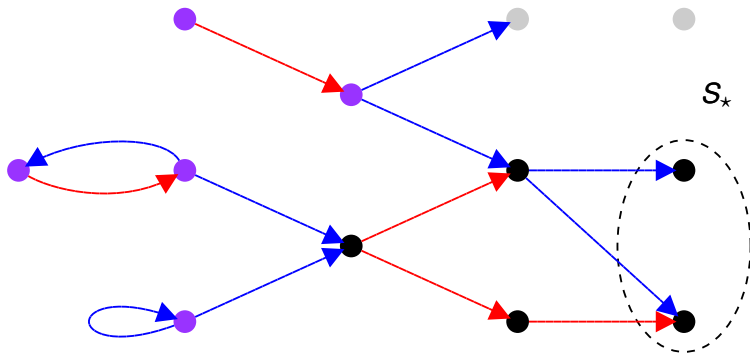


# Nested Fixpoint Algorithm

## Example



States from which goals are reachable in  $\leq 2$  steps so that all immediate successors are possibly good.



Strong cyclic plans

Motivation

**Nested Fixpoint Algorithm**

Incremental Planning Algorithm

Maintenance

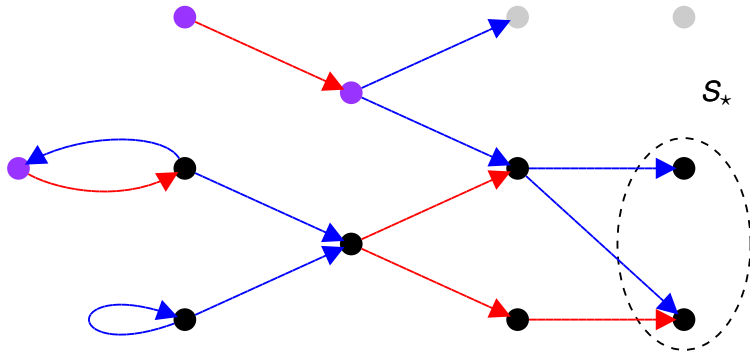
Summary

# Nested Fixpoint Algorithm

## Example



States from which goals are reachable in  $\leq 3$  steps so that all immediate successors are possibly good.



Strong cyclic  
plans

Motivation

Nested Fixpoint  
Algorithm

Incremental  
Planning Algorithm

Maintenance

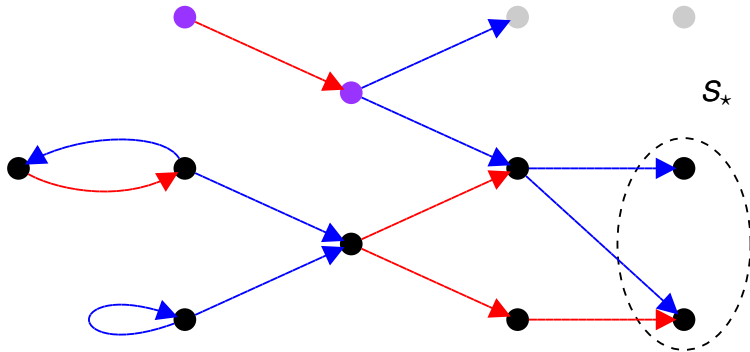
Summary

# Nested Fixpoint Algorithm

## Example



States from which goals are reachable in  $\leq 4$  steps so that all immediate successors are possibly good.



Strong cyclic  
plans

Motivation

**Nested Fixpoint  
Algorithm**

Incremental  
Planning Algorithm

Maintenance

Summary

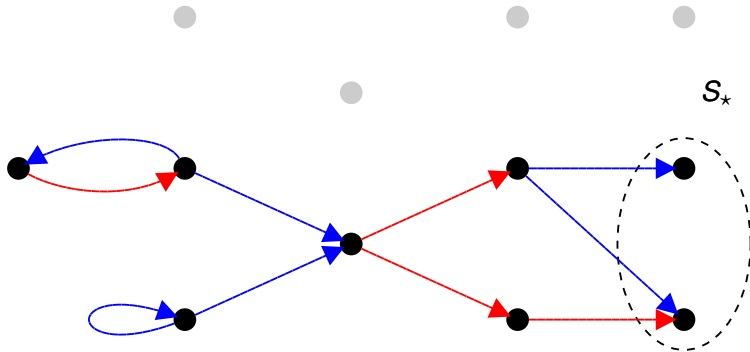
# Nested Fixpoint Algorithm

## Example



Remaining states are all good.

A further iteration would not eliminate more states.



Strong cyclic  
plans

Motivation

**Nested Fixpoint  
Algorithm**

Incremental  
Planning Algorithm

Maintenance

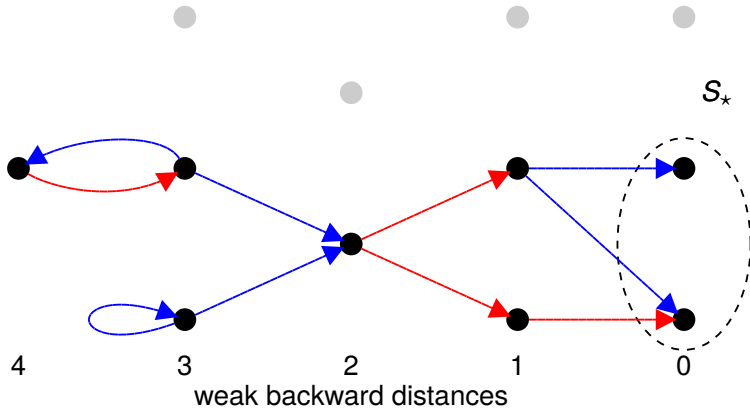
Summary

# Nested Fixpoint Algorithm

## Example



Assign each state an operator so that the successor states are goal states or good, and some of them are closer to goal states. Use **weak distances** computed with **weak preimages**. For this example this is trivial.



Strong cyclic plans

Motivation

Nested Fixpoint Algorithm

Incremental Planning Algorithm

Maintenance

Summary

Recall the definition of cyclic strong plans:

## Definition (strong cyclic plan)

Let  $S$  be the set of states of a planning task  $\Pi$ . Then a **strong cyclic plan** for  $\Pi$  is a function  $\pi : S_\pi \rightarrow O$  for some subset  $S_\pi \subseteq S$  such that

- $\pi(s)$  is applicable in  $s$  for all  $s \in S_\pi$ ,
- $S_\pi(s_0) \subseteq S_\pi \cup S_\star$  ( $\pi$  is closed), and
- $S_\pi(s') \cap S_\star \neq \emptyset$  for all  $s' \in S_\pi(s_0)$  ( $\pi$  is proper).

Strong cyclic plans

Motivation

Nested Fixpoint Algorithm

Incremental Planning Algorithm

Maintenance

Summary

- The procedure **prune** finds a maximal set of states for which reaching goals with looping is possible.
- It consists of two nested loops:
  - 1 The outer loop iterates through  $i = 0, 1, 2, \dots$  and produces a **shrinking** sequence of candidate good state sets  $C_0, C_1, \dots, C_n$  until  $C_i = C_{i+1}$ .
  - 2 The inner loop identifies **growing** sets  $W_j$  of states from which a goal state can be reached with  $j$  steps without leaving the current set of candidate good states  $C_i$ . The union of all  $W_0, W_1, \dots$  will be  $C_{i+1}$ .

Strong cyclic plans

Motivation

Nested Fixpoint Algorithm

Incremental Planning Algorithm

Maintenance

Summary

## Procedure *prune*

**def** *prune*( $S, O, S_*$ ):

$C_0 := S$

**for each**  $i \in \mathbb{N}_1$ :

$W_0 := S_*$

**for each**  $j \in \mathbb{N}_1$ :

$W_j := W_{j-1} \cup \bigcup_{o \in O} (\text{wpreimg}_o(W_{j-1}) \cap \text{spreimg}_o(C_{i-1}))$

**if**  $W_j = W_{j-1}$ :

**break**

$C_i := W_j$

**if**  $C_i = C_{i-1}$ :

**return**  $\langle C_i, \langle W_0, \dots, W_{j-1} \rangle \rangle$

Strong cyclic  
plans

Motivation

Nested Fixpoint  
Algorithm

Incremental  
Planning Algorithm

Maintenance

Summary



## Lemma (Procedure *prune*)

*Let  $S$  and  $S_\star \subseteq S$  be sets of states and  $O$  a set of operators. Then  $\text{prune}(S, O, S_\star)$  terminates after a finite number of steps and returns  $C \subseteq S$  such that there is a strategy  $\pi : C \setminus S_\star \rightarrow O$  that is a strong cyclic plan (for the states for which it is defined) and maximal in the sense that there is no set  $C' \supsetneq C$  and a strong cyclic plan  $\pi' : C' \setminus S_\star \rightarrow O$ .*

- The sets  $W_j$  also returned by *prune* encode weak distances and can be used to define the strong cyclic plan  $\pi$ .

Strong cyclic plans

Motivation

Nested Fixpoint Algorithm

Incremental Planning Algorithm

Maintenance

Summary

## The planning algorithm

**def** strong-cyclic-plan( $\langle V, I, O, \gamma \rangle$ ):

$S :=$  set of states over  $V$

$S_* := \{s \in S \mid s \models \gamma\}$

$\langle C, (W_j)_{j=0,1,2,\dots} \rangle = \text{prune}(S, O, S_*)$

**if**  $I \notin C$ :

**return** no solution

**for each**  $s \in C$ :

$\delta(s) := \min\{j \in \mathbb{N}_0 \mid s \in W_j\}$

**for each**  $s \in C \setminus S_*$ :

$\pi(s) :=$  some operator  $o \in O$  with  $\text{img}_o(s) \subseteq C$   
and  $\min\{\delta(s') \mid s' \in \text{img}_o(s)\} < \delta(s)$

**return**  $\pi$

Strong cyclic  
plans

Motivation

Nested Fixpoint  
Algorithm

Incremental  
Planning Algorithm

Maintenance

Summary

- 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 strong cyclic plans for full observability is in the complexity class EXPTIME.
- The problem is also EXPTIME-hard.
- Similar to strong planning, we can speed up the algorithm in many practical cases by using a symbolic implementation (e. g. with BDDs).

Strong cyclic plans

Motivation

Nested Fixpoint Algorithm

Incremental Planning Algorithm

Maintenance

Summary



**Idea** [Kuter/Nau/Reisner/Goldman, 2008; Fu/Ng/Bastiani/Yen, 2011]:

1. Pretend the planning task was deterministic: Turn each action  $o = \langle \chi, E \rangle$  with  $E = \{e_1, \dots, e_n\}$  into  $n$  actions  $o_i = \langle \chi, e_i \rangle$  for  $i = 1, \dots, n$ . Obtain classical problem  $\Pi'$ .
2. Find classical plan  $P$  in  $\Pi'$ . Add state-action mapping corresponding to  $P$  to  $\pi$ .
3. For each operator  $o_i$  used in  $P$  (in state  $s$ ), identify original nondeterministic operator  $o$  and states  $S' = \text{img}_o(s)$ .
4. For each “open” state  $s' \in S'$ , go to 2.

**Remark:** May require backtracking, if some state used in a classical plan turns out not to admit a strong cyclic plan.

Strong cyclic plans

Motivation

Nested Fixpoint Algorithm

Incremental Planning Algorithm

Maintenance

Summary



Strong cyclic  
plans

Motivation

Nested Fixpoint  
Algorithm

Incremental  
Planning Algorithm

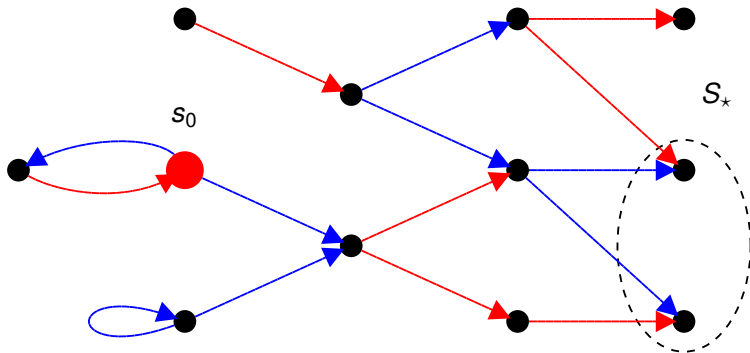
Maintenance

Summary

## Definition (all-outcomes determinization)

Let  $\Pi = \langle V, I, O, \gamma \rangle$  be a nondeterministic planning task. The **all-outcomes determinization** of  $\Pi$  is the deterministic planning task  $\Pi_{\text{det}} = \langle V, I, O_{\text{det}}, \gamma \rangle$ , where  $O_{\text{det}} = \bigcup_{o \in O} o_{\text{det}}$ , and  $\langle \chi, E \rangle_{\text{det}} = \{ \langle \chi, e \rangle \mid e \in E \}$ .

List of states to solve:  $\{s_0\}$



Strong cyclic  
plans

Motivation

Nested Fixpoint  
Algorithm

Incremental  
Planning Algorithm

Maintenance

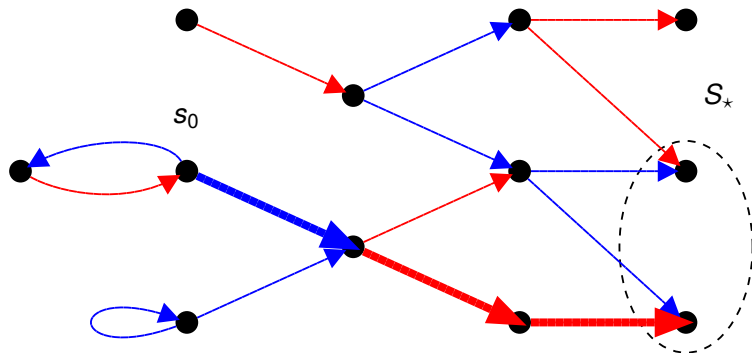
Summary

# Determinization-based Incremental Alg.

## Example



Plan for  $s_0$  in determinization: *blue<sub>2</sub>, red<sub>2</sub>, red*



Strong cyclic plans

Motivation

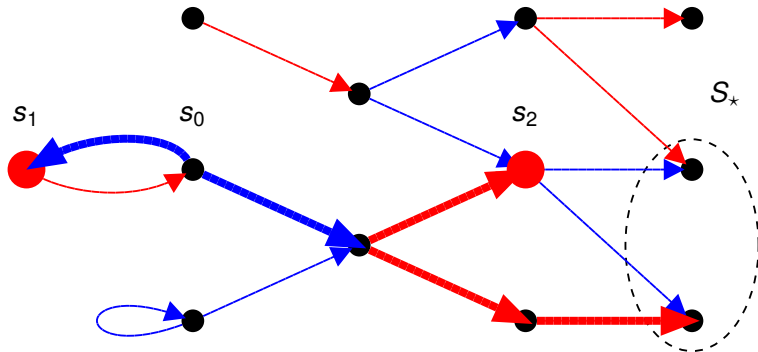
Nested Fixpoint Algorithm

Incremental Planning Algorithm

Maintenance

Summary

“Undesired” outcomes of  $blue_1$  and  $red_1$  lead to new list of states to solve:  $\{s_1, s_2\}$



Strong cyclic  
plans

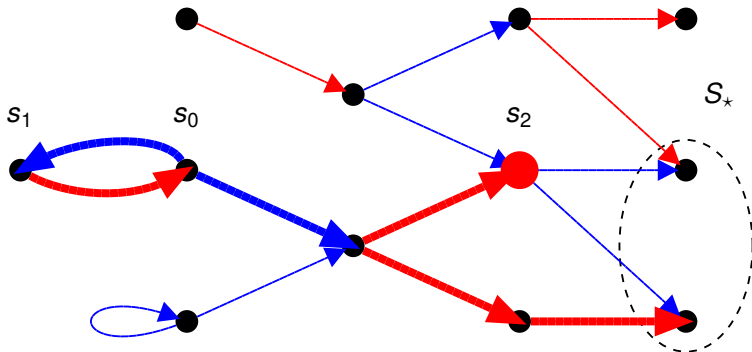
Motivation  
Nested Fixpoint  
Algorithm  
Incremental  
Planning Algorithm

Maintenance

Summary



Plan for  $s_1$  in determinization: *red*, *blue*<sub>2</sub>, *red*<sub>2</sub>, *red*



Strong cyclic plans

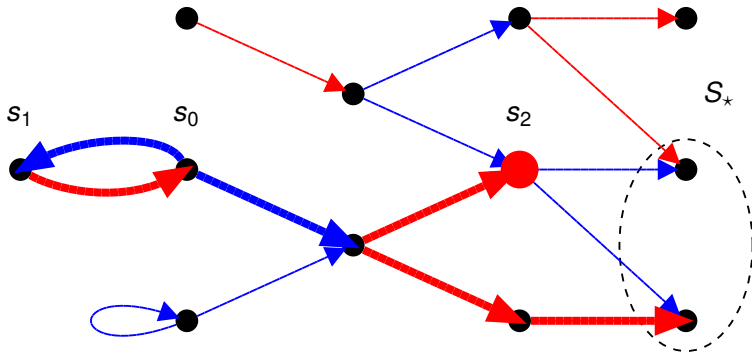
Motivation  
Nested Fixpoint Algorithm  
Incremental Planning Algorithm

Maintenance

Summary

No new “undesired” outcomes.

List of states to solve:  $\{s_2\}$



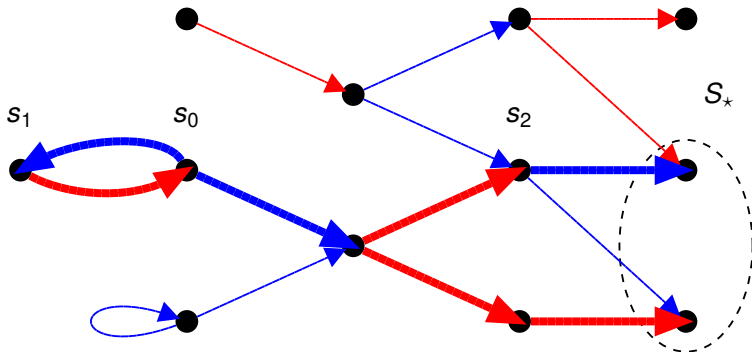
Strong cyclic plans

Motivation  
Nested Fixpoint Algorithm  
Incremental Planning Algorithm

Maintenance

Summary

Plan for  $s_2$  in determinization: *blue<sub>1</sub>*



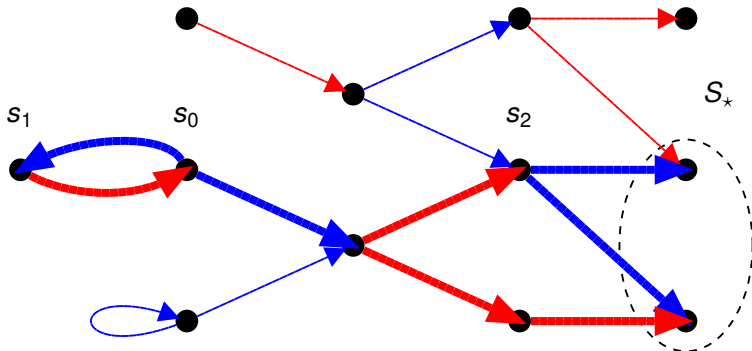
Strong cyclic  
plans

Motivation  
Nested Fixpoint  
Algorithm  
Incremental  
Planning Algorithm

Maintenance

Summary

“Undesired” outcome of *blue*<sub>2</sub> in *s*<sub>2</sub> leads to goal state, too.  
List of states to solve:  $\emptyset$ . Strong cyclic plan found.



Strong cyclic  
plans

Motivation  
Nested Fixpoint  
Algorithm  
Incremental  
Planning Algorithm

Maintenance

Summary



## Procedure incremental-strong-cyclic-plan

**def** incremental-strong-cyclic-plan( $\langle V, I, O, \gamma \rangle$ ):

$\pi \leftarrow \emptyset$ ; *fail*  $\leftarrow \{I\}$

**while** *fail*  $\neq \emptyset$ :

$s \leftarrow \text{SELECTANDREMOVEFROM}(\textit{fail})$

$\pi' \leftarrow \text{DETSEARCH}(\langle V, s, O_{\text{det}}, \gamma \rangle)$

**if**  $\pi' = \text{FAILURE}$ :

**if**  $s = I$ : **return** FAILURE

**else**:  $\text{BACKTRACK}(s, \pi, \langle V, I, O, \gamma \rangle)$

**else**:

$\pi \leftarrow \pi \cup \pi'$

*fail*  $\leftarrow \{s \in S \mid s \text{ nongoal state reachable from } I$   
following  $\pi$ , but  $\pi(s)$  undefined}

**return**  $\pi$

Strong cyclic  
plans

Motivation

Nested Fixpoint  
Algorithm

Incremental  
Planning Algorithm

Maintenance

Summary

If a deterministic search fails, the state  $s$  from which it started cannot be part of a strong cyclic plan.

- If  $s = l$ , the whole given planning problem is unsolvable and the algorithm returns FAILURE.
- Otherwise, state  $s$ , which has already been added to the constructed policy  $\pi$ , has to be removed from  $\pi$ , and the algorithm has to ensure that  $s$  will never be reconsidered again. This is accomplished by the procedure BACKTRACK.

Strong cyclic plans

Motivation

Nested Fixpoint Algorithm

Incremental Planning Algorithm

Maintenance

Summary



## Procedure backtrack

**def** backtrack( $s, \pi, \langle V, I, O, \gamma \rangle$ ):

update  $\pi$  by deleting all entries that would immediately

lead to  $s$ , i.e.  $\pi \leftarrow \pi \setminus \{(s', \pi(s')) \mid s \in \text{img}_{\pi(s')}(s')\}$

add all states  $s'$  removed from  $\pi$  to the set of fail-states *fail*

permanently mark all formerly assigned actions  $\pi(s')$

removed from  $\pi$  at  $s'$  as inapplicable in  $s'$  to avoid

running into the same dead end again.

Strong cyclic  
plans

Motivation

Nested Fixpoint  
Algorithm

Incremental  
Planning Algorithm

Maintenance

Summary



- Iteratively solves all-outcomes determinizations of  $\Pi$  with “fail-states” as initial states.
- Planner can choose desired outcome of each action.
- Deterministic plans are added to policy under construction.
- Corresponding undesired outcomes have to be added to the set of “fail-states” *fail*.
- Deterministic plans for “fail-states” are constructed until no more “fail-states” remain.
- Eventually, the algorithm either returns a strong cyclic plan or FAILURE if no such plan exists.

Strong cyclic plans

Motivation

Nested Fixpoint Algorithm

Incremental Planning Algorithm

Maintenance

Summary





## Theorem

*Procedure incremental-strong-cyclic-plan, called with task  $\Pi$ , returns a strong cyclic plan for  $\Pi$  iff such a plan exists, and FAILURE, otherwise.*

Strong cyclic plans

Motivation

Nested Fixpoint Algorithm

Incremental Planning Algorithm

Maintenance

Summary

# Determinization-based Incremental Alg.

Underlying classical planner



UNI  
FREIBURG

- Can use any classical planner for deterministic searches.
- Can benefit from heuristics etc. used there.
- Classical planner can be configured to prefer short solutions or solutions using deterministic actions induced by nondeterministic actions with few different outcomes (likely fewer new “fail-states”).

Strong cyclic plans

Motivation

Nested Fixpoint Algorithm

Incremental Planning Algorithm

Maintenance

Summary



- When to terminate a deterministic sub-search?
  - At goal states?
  - At states currently part of the partial solution?
  - At parent of currently solved “fail-state”?

This can make a huge difference.

- Similarly: Where should the heuristic guide the classical planner? Goals, partial solution, parent node?
- Additional marking of nodes as definitely solved if this can be detected.
- State reuse between subsequent classical planner calls.
- Generalization of solved states by regression search from goal along weak (deterministic) plan (cf. [Muisse/McIlraith/Beck, 2012]).

Strong cyclic plans

Motivation

Nested Fixpoint Algorithm

Incremental Planning Algorithm

Maintenance

Summary

## 2 Maintenance goals

- Definition
- Example
- Algorithm

Strong cyclic  
plans

### Maintenance

Definition

Example

Algorithm

Summary



- In this lecture, we usually limit ourselves to the problem of finding plans that **reach a goal state**.
- In practice, planning is often about more general goals, where execution cannot be terminated.
  - 1 An animal: find food, eat, sleep, find food, eat, sleep, ...
  - 2 Cleaning robot: keep the building clean.
- These problems cannot be directly formalized in terms of reachability because infinite (unbounded) plan execution is needed.
- We do not discuss this topic in full detail. However, to give at least a little impression of **planning for temporally extended goals**, we will discuss the simplest objective with infinite plan executions: **maintenance**.

Strong cyclic plans

Maintenance

Definition

Example

Algorithm

Summary

## Definition

Let  $\mathcal{T} = \langle V, I, O, \gamma \rangle$  be a planning task with state set  $S$  and set of goal states  $S_\star = \{s \in S \mid s \models \gamma\}$ .

A strategy  $\pi$  for  $\mathcal{T}$  is called a **plan for maintenance** for  $\mathcal{T}$  iff

- $\pi(s)$  is applicable in  $s$  for all  $s \in S_\pi$ ,
- $S_\pi(s_0) \subseteq S_\pi$ , and
- $S_\pi(s_0) \subseteq S_\star$ .

- The state of an animal is determined by three state values: 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 avoid death.

Strong cyclic plans

Maintenance

Definition

Example

Algorithm

Summary

# Maintenance goals

Transition system for the example 0-safe states 1-safe states  $i$ -safe states for all  $i \geq 2$



Strong cyclic plans

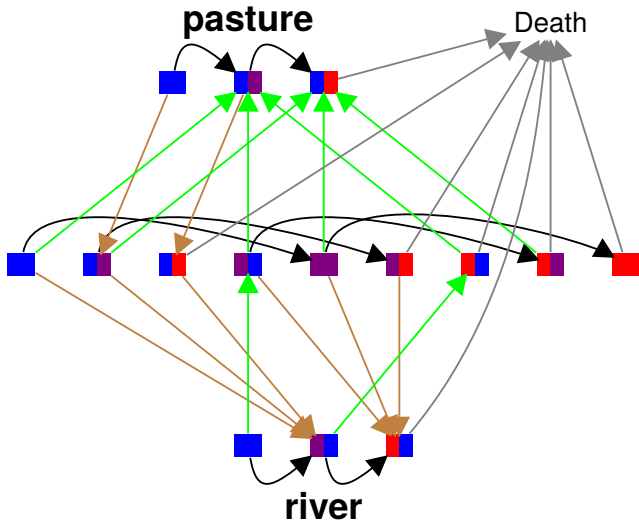
Maintenance

Definition

Example

Algorithm

Summary





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.

If the above rules conflict, the animal will die.

Strong cyclic plans

Maintenance

Definition

Example

Algorithm

Summary

## Summary of the algorithm idea

Repeatedly eliminate from consideration those states that in one or more steps unavoidably lead to a non-goal state.

- A state is *i-safe* iff there is a plan that guarantees “survival” for the next  $i$  actions.
- A state is *safe* (or  $\infty$ -safe) iff it is  $i$ -safe for all  $i \in \mathbb{N}_0$ .
- The *0-safe* states are exactly the goal states: maintenance objective is satisfied for the current state.
- Given all  $i$ -safe states, compute all  $i + 1$ -safe states by using strong preimages.
- For some  $i \in \mathbb{N}_0$ ,  $i$ -safe states equal  $i + 1$ -safe states because there are only finitely many states and at each step and  $i + 1$ -safe states are a subset of  $i$ -safe states. Then  $i$ -safe states are also  $\infty$ -safe.

Strong cyclic plans

Maintenance

Definition

Example

Algorithm

Summary

## Planning for maintenance goals

**def** maintenance-plan( $\langle V, I, O, \gamma \rangle$ ):

$S :=$  set of states over  $V$

$Safe_0 := \{s \in S \mid s \models \gamma\}$

**for each**  $i \in \mathbb{N}_1$ :

$Safe_i := Safe_{i-1} \cap \bigcup_{o \in O} spreimg_o(Safe_{i-1})$

**if**  $Safe_i = Safe_{i-1}$ :

**break**

**if**  $I \notin Safe_i$ :

**return** no solution

**for each**  $s \in Safe_i$ :

$\pi(s) :=$  some operator  $o \in O$  with  $img_o(s) \subseteq Safe_i$

**return**  $\pi$

Strong cyclic  
plans

Maintenance

Definition

Example

Algorithm

Summary

# Maintenance goals

Transition system for the example 0-safe states 1-safe states  $i$ -safe states for all  $i \geq 2$



Strong cyclic plans

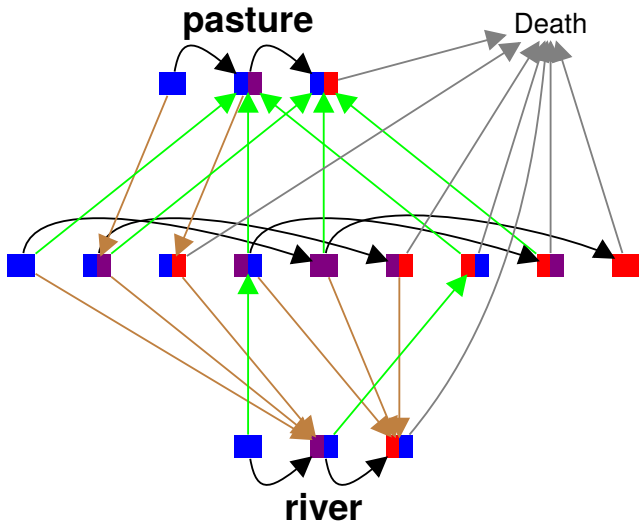
Maintenance

Definition

Example

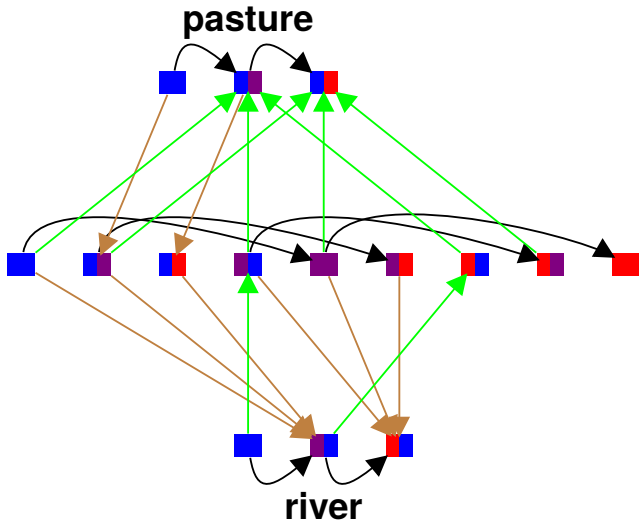
Algorithm

Summary



# Maintenance goals

Transition system for the example 0-safe states 1-safe states  $i$ -safe states for all  $i \geq 2$



Strong cyclic  
plans

Maintenance

Definition

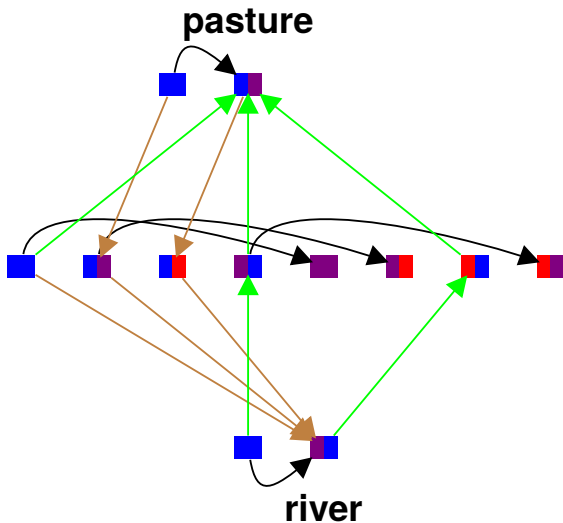
Example

Algorithm

Summary

# Maintenance goals

Transition system for the example 0-safe states 1-safe states  $i$ -safe states for all  $i \geq 2$



Strong cyclic plans

Maintenance

Definition

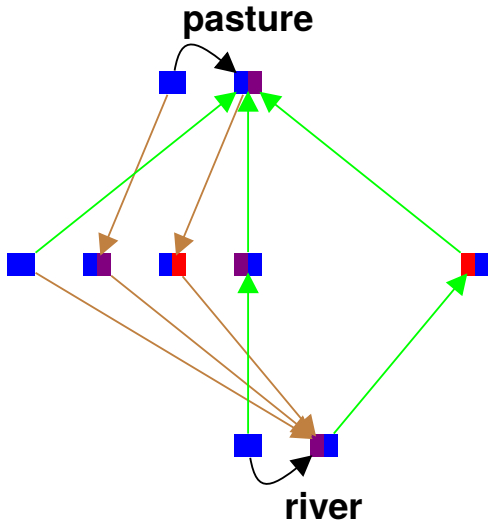
Example

Algorithm

Summary

# Maintenance goals

Transition system for the example 0-safe states 1-safe states  $i$ -safe states for all  $i \geq 2$



Strong cyclic plans

Maintenance

Definition

Example

Algorithm

Summary

# 3 Summary



Strong cyclic  
plans

Maintenance

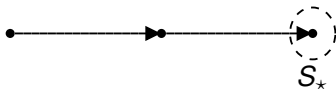
Summary



# Different planning objectives

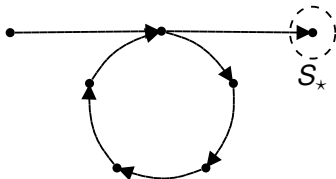


Strong planning



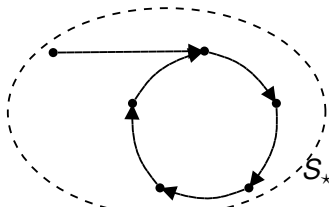
Strong cyclic plans

Strong cyclic planning



Maintenance

Maintenance



Summary



- We have considered different classes of solutions for planning tasks by defining **different planning problems**.
  - strong planning problem: find a strong plan
  - strong cyclic planning problem: find a strong cyclic plan
  - ...
- Alternatively, we could allow specifying goals in a **modal logic** like **computational tree logic** to directly express the type of plan we are interested in using **modalities** such as A (all), E (exists), G (globally), and F (finally).
  - Weak planning:  $EF\varphi$
  - Strong planning:  $AF\varphi$
  - Strong cyclic planning:  $AGEF\varphi$
  - Maintenance:  $AG\varphi$

Strong cyclic plans

Maintenance

Summary



- We have extended our earlier planning algorithm from **strong** plans to **strong cyclic** plans.
- The story does not end there: When considering infinitely executing plans, many more types of goals are feasible.
- We considered **maintenance** as a simple example of a **temporally extended goal**.
- In general, temporally extended goals be expressed in **modal logics** such as computational tree logic (CTL).
- We presented dynamic programming (backward search) algorithms for strong cyclic and maintenance planning.
- In practice, one might implement both algorithms by using binary decision diagrams (BDDs) as a data structure for state sets.

Strong cyclic  
plans

Maintenance

Summary