Bounded Branching and Modalities in Non-Deterministic Planning

Blai Bonet

Departamento de Computación

Universidad Simón Boívar

bonet@ldc.usb.ve

Introduction

- We consider two extensions for the task of deciding the existence of solutions for non-deterministic planning problems:
    - Bounds in the number of branch points in a plan (solution)
    - Extending the description language with modal formulae
- The first applies to the cases of non-deterministic planning with complete and partial information
- The second applies only to the case of non-deterministic planning with partial information

Introduction

- We consider two extensions for the task of deciding the existence of solutions for non-deterministic planning problems:
  - Bounds in the number of branch points in a plan (solution)
  - Extending the description language with modal formulae
- The first applies to the cases of non-deterministic planning with complete and partial information
- The second applies only to the case of non-deterministic planning with partial information

Introduction

- We consider two extensions for the task of deciding the existence of solutions for non-deterministic planning problems:
  - Bounds in the number of branch points in a plan (solution)
  - Extending the description language with modal formulae
- The first applies to the cases of non-deterministic planning with complete and partial information
- The second applies only to the case of non-deterministic planning with partial information

Outline

- Planning with Complete Information
- Planning with Partial Information
- Summary

Planning with Complete Information

Introduction

- Non-deterministic planning deals with planning problems with actions that might have more than one outcome (non-deterministic actions)

Introduction

- Non-deterministic planning deals with planning problems with actions that might have more than one outcome (non-deterministic actions)

- After the application of such an action, the agent **observes** the resulting state of the system and chooses the next action

Introduction

- Non-deterministic planning deals with planning problems with actions that might have more than one outcome (non-deterministic actions)
- After the application of such an action, the agent **observes** the resulting state of the system and chooses the next action
- This constitutes a **branch point** in the plan

Introduction

- Non-deterministic planning deals with planning problems with actions that might have more than one outcome (non-deterministic actions)
- After the application of such an action, the agent **observes** the resulting state of the system and chooses the next action
- This constitutes a **branch point** in the plan
- Another possibility is to apply a sequence of actions without observing the system and then to observe the state of the system at the end of the sequence and planning thereafter

Introduction

- Non-deterministic planning deals with planning problems with actions that might have more than one outcome (non-deterministic actions)
- After the application of such an action, the agent **observes** the resulting state of the system and chooses the next action
- This constitutes a **branch point** in the plan
- Another possibility is to apply a sequence of actions without observing the system and then to observe the state of the system at the end of the sequence and planning thereafter
- Therefore, we can think of bounding the number of branch points in a plan and to question when such a plan exists

Deterministic Models

Understood in terms of:

- a discrete and finite state space S
- an initial state $s_0 \in S$
- a non-empty set of goal states $G \subseteq S$
- actions $A(s) \subseteq A$ applicable in each state s
- a function that maps states and actions into states $f(a,s) \in S$

Deterministic Models

Understood in terms of:

- a discrete and finite state space S
- an initial state $s_0 \in S$
- a non-empty set of goal states $G \subseteq S$
- actions $A(s) \subseteq A$ applicable in each state s
- a function that maps states and actions into states $f(a,s) \in S$

Solutions: sequences $(a_0, \ldots, a_n)$ of actions that “transform” s_0 into a goal state

Description Language

- Propositional language used to compactly describe the transition function $f(\cdot, \cdot)$ and the applicable actions $A(\cdot)$

Description Language

- Propositional language used to compactly describe the transition function $f(\cdot,\cdot)$ and the applicable actions $A(\cdot)$
- States are valuations to propositional symbols

Description Language

- Propositional language used to compactly describe the transition function $f(\cdot,\cdot)$ and the applicable actions $A(\cdot)$
- States are valuations to propositional symbols
- Usually represented as the set of propositions that hold true in the state

Description Language

- Propositional language used to compactly describe the transition function $f(\cdot,\cdot)$ and the applicable actions $A(\cdot)$
- States are valuations to propositional symbols
- Usually represented as the set of propositions that hold true in the state
- We use an action language similar to that in [Rintanen, 2004]:
  - Actions are pairs $\langle prec, effect \rangle$
  - $prec$ is a propositional formula used to define $A(s)$
  - Effects are defined inductively as:
    - $\{l\}$ is an effect for literal l (atomic)
    - $(e_1 \wedge \ldots \wedge e_n)$ for effects $e_1, \ldots, e_n$ (parallel)
    - $(c \triangleright e)$ for effect e and formula c (conditional)

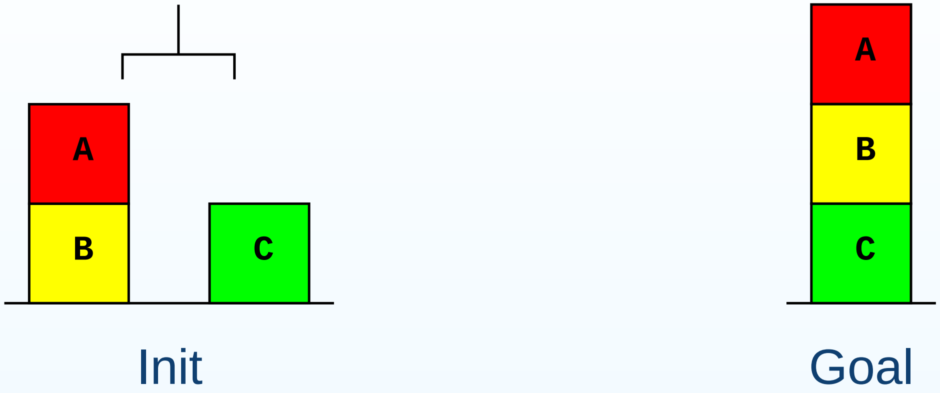
Description Language

- Propositional language used to compactly describe the transition function $f(\cdot,\cdot)$ and the applicable actions $A(\cdot)$
- States are valuations to propositional symbols
- Usually represented as the set of propositions that hold true in the state
- We use an action language similar to that in [Rintanen, 2004]:
  - Actions are pairs $\langle prec, effect \rangle$
  - $prec$ is a propositional formula used to define $A(s)$
  - Effects are defined inductively as:
    - $\{l\}$ is an effect for literal l (atomic)
    - $(e_1 \wedge \ldots \wedge e_n)$ for effects $e_1, \ldots, e_n$ (parallel)
    - $(c \triangleright e)$ for effect e and formula c (conditional)
- The initial state defined by the set I of propositions that hold true

Description Language

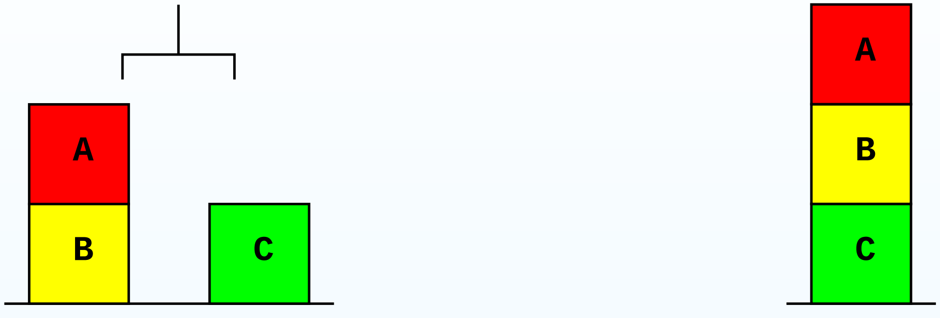
- Propositional language used to compactly describe the transition function $f(\cdot,\cdot)$ and the applicable actions $A(\cdot)$
- States are valuations to propositional symbols
- Usually represented as the set of propositions that hold true in the state
- We use an action language similar to that in [Rintanen, 2004]:
  - Actions are pairs $\langle prec, effect \rangle$
  - $prec$ is a propositional formula used to define $A(s)$
  - Effects are defined inductively as:
    - $\{l\}$ is an effect for literal l (atomic)
    - $(e_1 \wedge \ldots \wedge e_n)$ for effects $e_1, \ldots, e_n$ (parallel)
    - $(c \rhd e)$ for effect e and formula c (conditional)
- The initial state defined by the set I of propositions that hold true
- The goal states defined by a propositional formula Φ_G

Example – Blocksworld (Deterministic)



- Propositions:
  - Blocks' positions: $\{$`on-table(B)`, `on(A,B)`, `on-table(C)`$\}$
  - Others: $\{$`clear(A)`, `clear(C)`, `empty-hand`$\}$

Example – Blocksworld (Deterministic)

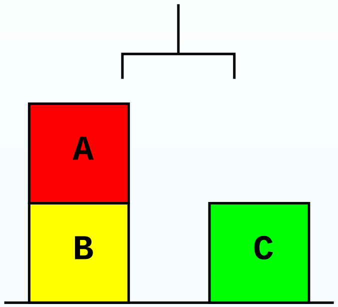


Init Goal

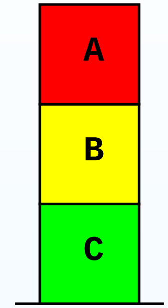
- Propositions:
  - Blocks' positions: $\{\texttt{on-table(B)}, \texttt{on(A,B)}, \texttt{on-table(C)}\}$
  - Others: $\{\texttt{clear(A)}, \texttt{clear(C)}, \texttt{empty-hand}\}$

- Actions:
  - **unstack(A,B)**:
    $\langle \texttt{empty-hand} \wedge \texttt{clear(A)} \wedge \texttt{on(A,B)}, \texttt{holding(A)} \wedge \texttt{clear(B)} \wedge \neg\texttt{on(A,B)} \rangle$
  - **pick(A)**: $\langle \texttt{empty-hand} \wedge \texttt{clear(A)} \wedge \texttt{on-table(A)}, \texttt{holding(A)} \wedge \neg\texttt{on-table(A)} \rangle$
  - **stack(A,B)**: $\langle \texttt{holding(A)} \wedge \texttt{clear(B)}, \texttt{empty-hand} \wedge \texttt{on(A,B)} \wedge \neg\texttt{holding(A)} \rangle$
  - **drop(A)**: $\langle \texttt{holding(A)}, \texttt{empty-hand} \wedge \texttt{on-table(A)} \wedge \neg\texttt{holding(A)} \rangle$

Example – Blocksworld (Deterministic)



Init



Goal

- Propositions:
  - Blocks' positions: $\{\texttt{on-table(B)}, \texttt{on(A,B)}, \texttt{on-table(C)}\}$
  - Others: $\{\texttt{clear(A)}, \texttt{clear(C)}, \texttt{empty-hand}\}$

- Actions:
  - **unstack(A,B)**:

    $\langle \texttt{empty-hand} \wedge \texttt{clear(A)} \wedge \texttt{on(A,B)}, \texttt{holding(A)} \wedge \texttt{clear(B)} \wedge \neg\texttt{on(A,B)} \rangle$
  - **pick(A)**: $\langle \texttt{empty-hand} \wedge \texttt{clear(A)} \wedge \texttt{on-table(A)}, \texttt{holding(A)} \wedge \neg\texttt{on-table(A)} \rangle$
  - **stack(A,B)**: $\langle \texttt{holding(A)} \wedge \texttt{clear(B)}, \texttt{empty-hand} \wedge \texttt{on(A,B)} \wedge \neg\texttt{holding(A)} \rangle$
  - **drop(A)**: $\langle \texttt{holding(A)}, \texttt{empty-hand} \wedge \texttt{on-table(A)} \wedge \neg\texttt{holding(A)} \rangle$

- Plan: $(\texttt{unstack(A,B)}, \texttt{drop(A)}, \texttt{pick(B)}, \texttt{stack(B,C)}, \texttt{pick(A)}, \texttt{stack(A,B)})$

Complexity of Deterministic Planning

- The existence of a plan can be decided with the non-deterministic program:

  1. Let $counter := 0$
  2. Let $state := I$
  3. If $state \models \Phi_G$, then ACCEPT
  4. Choose applicable action a in $state$
  5. Let $state := f(a, state)$
  6. Let $counter := counter + 1$
  7. If $counter = 2^{|P|}$, then REJECT
  8. Goto 3

Complexity of Deterministic Planning

- The existence of a plan can be decided with the non-deterministic program:

    1. Let $counter := 0$
    2. Let $state := I$
    3. If $state \models \Phi_G$, then ACCEPT
    4. Choose applicable action a in $state$
    5. Let $state := f(a, state)$
    6. Let $counter := counter + 1$
    7. If $counter = 2^{|P|}$, then REJECT
    8. Goto 3

- Therefore, PLAN-DET is in NPSPACE = PSPACE

Complexity of Deterministic Planning

- The existence of a plan can be decided with the non-deterministic program:

  1. Let $counter := 0$
  2. Let $state := I$
  3. If $state \models \Phi_G$, then ACCEPT
  4. Choose applicable action a in $state$
  5. Let $state := f(a, state)$
  6. Let $counter := counter + 1$
  7. If $counter = 2^{|P|}$, then REJECT
  8. Goto 3

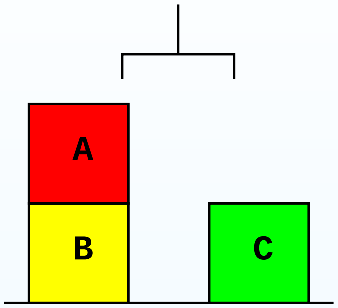
- Therefore, PLAN-DET is in NPSPACE = PSPACE

- The fact than PLAN-DET is PSPACE-hard was shown in [Bylander, 1994] with a direct simulation of DTMs with polynomial space bound

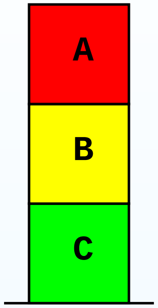
Non-Deterministic Models

- As deterministic models but the transition function maps states and actions into **sets of states** $F(a,s) \subseteq S$
- There can be more than one initial state described by formula Φ_I
- The description language is extended with non-deterministic effects:
  - $(e_1 \oplus \cdots \oplus e_n)$ for effects $e_1,\ldots,e_n$ (non-deterministic effect),
- Solutions can't be sequences of actions, but tree-like structures called **contingent plans**

Example – Blocksworld (Non-Deterministic)



Init

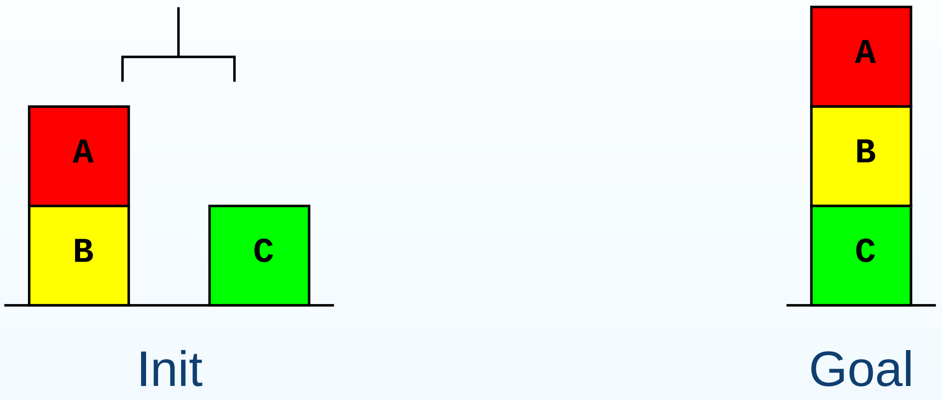


Goal

- New Action:
  - **unstack(A,B)**:

$$\langle \texttt{empty-hand} \wedge \texttt{clear(A)} \wedge \texttt{on(A,B)},$$
$$(\texttt{holding(A)} \wedge \texttt{clear(B)} \wedge \neg\texttt{on(A,B)}) \oplus$$
$$(\texttt{clear(B)} \wedge \texttt{on-table(A)} \wedge \neg\texttt{on(A,B)})\rangle$$

Example – Blocksworld (Non-Deterministic)



- New Action:
  - **unstack(A,B)**:

$$\langle \texttt{empty-hand} \wedge \texttt{clear(A)} \wedge \texttt{on(A,B)}, \\ (\texttt{holding(A)} \wedge \texttt{clear(B)} \wedge \neg \texttt{on(A,B)}) \oplus \\ (\texttt{clear(B)} \wedge \texttt{on-table(A)} \wedge \neg \texttt{on(A,B)}) \rangle$$

- Contingent Plan:

$$\texttt{unstack(A,B)} \begin{cases} \texttt{drop(A)}, \texttt{pick(B)}, \texttt{stack(B,C)}, \texttt{pick(A)}, \texttt{stack(A,B)} \\ \texttt{pick(B)}, \texttt{stack(B,C)}, \texttt{pick(A)}, \texttt{stack(A,B)} \end{cases}$$

Complexity of Non-Deterministic Planning

- Plans are like policies for Markov Decisions Processes (MDPs)
- Deciding existence of solution for a contingent planning problem with full observability (i.e. PLAN-FO-CONT) is EXPTIME-complete
- Shown by [Rintanen, 2004] using Alternating TMs with polynomial space bound
- Interestingly, [Littman, 1997] showed that deciding the existence of acyclic policies, of bounded depth N, that reach the goal with probability $\geq T$ for MDPs is also EXPTIME-complete

Conformant Planning

- Consider now the following actions:
  - **unstack(A,B)**:
    $\langle true, (\texttt{empty-hand} \wedge \texttt{clear(A)} \wedge \texttt{on(A,B)} \rhd \texttt{holding(A)} \wedge \texttt{clear(B)} \wedge \neg\texttt{on(A,B)})\rangle$
  - **pick(A)**:
    $\langle true, (\texttt{empty-hand} \wedge \texttt{clear(A)} \wedge \texttt{on-table(A)} \rhd \texttt{holding(A)} \wedge \neg\texttt{on-table(A)})\rangle$
  - **stack(A,B)**:
    $\langle true, (\texttt{holding(A)} \wedge \texttt{clear(B)} \rhd \texttt{empty-hand} \wedge \texttt{on(A,B)} \wedge \neg\texttt{holding(A)})\rangle$
  - **drop(A)**:
    $\langle true, (\texttt{holding(A)} \rhd \texttt{empty-hand} \wedge \texttt{on-table(A)} \wedge \neg\texttt{holding(A)})\rangle$

Conformant Planning

- Consider now the following actions:
  - **unstack(A,B)**:
    $\langle \mathit{true}, (\texttt{empty-hand} \land \texttt{clear(A)} \land \texttt{on(A,B)} \rhd \texttt{holding(A)} \land \texttt{clear(B)} \land \neg\texttt{on(A,B)}) \rangle$
  - **pick(A)**:
    $\langle \mathit{true}, (\texttt{empty-hand} \land \texttt{clear(A)} \land \texttt{on-table(A)} \rhd \texttt{holding(A)} \land \neg\texttt{on-table(A)}) \rangle$
  - **stack(A,B)**:
    $\langle \mathit{true}, (\texttt{holding(A)} \land \texttt{clear(B)} \rhd \texttt{empty-hand} \land \texttt{on(A,B)} \land \neg\texttt{holding(A)}) \rangle$
  - **drop(A)**:
    $\langle \mathit{true}, (\texttt{holding(A)} \rhd \texttt{empty-hand} \land \texttt{on-table(A)} \land \neg\texttt{holding(A)}) \rangle$

- Like the deterministic case but with **no preconditions** and **conditional effects**

Conformant Planning

- Consider now the following actions:
  - **`unstack(A,B)`**:
    $\langle true, (\texttt{empty-hand} \wedge \texttt{clear(A)} \wedge \texttt{on(A,B)} \rhd \texttt{holding(A)} \wedge \texttt{clear(B)} \wedge \neg\texttt{on(A,B)})\rangle$
  - **`pick(A)`**:
    $\langle true, (\texttt{empty-hand} \wedge \texttt{clear(A)} \wedge \texttt{on-table(A)} \rhd \texttt{holding(A)} \wedge \neg\texttt{on-table(A)})\rangle$
  - **`stack(A,B)`**:
    $\langle true, (\texttt{holding(A)} \wedge \texttt{clear(B)} \rhd \texttt{empty-hand} \wedge \texttt{on(A,B)} \wedge \neg\texttt{holding(A)})\rangle$
  - **`drop(A)`**:
    $\langle true, (\texttt{holding(A)} \rhd \texttt{empty-hand} \wedge \texttt{on-table(A)} \wedge \neg\texttt{holding(A)})\rangle$

- Like the deterministic case but with **no preconditions** and **conditional effects**

- Since there are no preconditions, the actions are **always applicable**

Conformant Planning (Cont'd) and Complexity

- Therefore, the plan:

`pick(A)`, `drop(A)`, `pick(B)`, `drop(B)`, `pick(C)`, `drop(C)`,

`pick(A)`, `drop(A)`, `pick(B)`, `drop(B)`, `pick(C)`, `drop(C)`,

`pick(B)`, `stack(B,C)`, `pick(A)`, `stack(A,B)`

achieves the goal (i.e. `A on B on C`) **no matter what's the initial situation**

Conformant Planning (Cont'd) and Complexity

- Therefore, the plan:

  `pick(A), drop(A), pick(B), drop(B), pick(C), drop(C),`

  `pick(A), drop(A), pick(B), drop(B), pick(C), drop(C),`

  `pick(B), stack(B,C), pick(A), stack(A,B)`

  achieves the goal (i.e. `A on B on C`) **no matter what's the initial situation**

- This plan is called a **conformant plan** [Goldman & Boddy, 1996; Smith & Weld, 1998]

Conformant Planning (Cont'd) and Complexity

- Therefore, the plan:

$$\texttt{pick(A)},\texttt{drop(A)},\texttt{pick(B)},\texttt{drop(B)},\texttt{pick(C)},\texttt{drop(C)},$$

$$\texttt{pick(A)},\texttt{drop(A)},\texttt{pick(B)},\texttt{drop(B)},\texttt{pick(C)},\texttt{drop(C)},$$

$$\texttt{pick(B)},\texttt{stack(B,C)},\texttt{pick(A)},\texttt{stack(A,B)}$$

  achieves the goal (i.e. `A on B on C`) **no matter what's the initial situation**

- This plan is called a **conformant plan** [Goldman & Boddy, 1996; Smith & Weld, 1998]

- A conformant plan is a **no-branch** plan for a non-deterministic planning problem with full observability

Complexity of Conformant Planning

- Checking the existence of a conformant plan (i.e. PLAN-FO-CONF) is EXPSPACE-complete
- The inclusion is shown with a similar program that works with subsets of states instead of states
- Hardness shown by [Haslum & Jonsson, 1999] using Regular Expressions with Exponentiation and Non-deterministic Finite Automata with Counters

Complexity of Conformant Planning

- Checking the existence of a conformant plan (i.e. PLAN-FO-CONF) is EXPSPACE-complete
- The inclusion is shown with a similar program that works with subsets of states instead of states
- Hardness shown by [Haslum & Jonsson, 1999] using Regular Expressions with Exponentiation and Non-deterministic Finite Automata with Counters

- Example: Σ^n set of words of length n over alphabet Σ
- An REE α can be recognized with an NFA with Counters (NFAC)
- The NFAC can be "simulated" as a non-deterministic planning problem P such that $\alpha \neq \Sigma^*$ iff P has a conformant plan
- Therefore, since checking whether $\alpha = \Sigma^*$ is EXPSPACE-hard, checking if P has a conformant plan is co-EXPSPACE-hard
- Finish with the fact that EXPSPACE is closed under complementation

Plans of Bounded Branching

- Contingent and conformant planning are the extreme points of a discrete yet infinite range of solution forms:

Plans of Bounded Branching

- Contingent and conformant planning are the extreme points of a discrete yet infinite range of solution forms:
  - Conformant = No branch
  - Contingent = Unbounded branch

Plans of Bounded Branching

- Contingent and conformant planning are the extreme points of a discrete yet infinite range of solution forms:
  - Conformant = No branch
  - Contingent = Unbounded branch
- In the middle, we can think of plans with no more than k branches

Plans of Bounded Branching

- Contingent and conformant planning are the extreme points of a discrete yet infinite range of solution forms:
  - Conformant = No branch
  - Contingent = Unbounded branch
- In the middle, we can think of plans with no more than k branches
- Checking the existence of a contingent plan with at most k branches (i.e. PLAN-FO-CONT-k) is EXPSPACE-complete
- Similar proof to that of [Haslum & Jonsson, 1999] for conformant planning

Summary

Problem	Complete for	Reference
PLAN-DET	PSPACE	[Bylander, 1994]
PLAN-FO-CONT	EXPTIME	[Rintanen, 2004]
PLAN-FO-CONF	EXPSPACE	[Haslum & Jonsson, 1999]
PLAN-FO-CONT-k	EXPSPACE	**New**

Remember:

- P $\subseteq$ NP $\subseteq$ PSPACE $\subseteq$ EXPTIME $\subseteq$ EXPSPACE
- PSPACE $= \cup_{k\geq 0}$DSPACE(n^k)
- EXPTIME $= \cup_{k\geq 0}$DTIME(2^{n^k})
- EXPSPACE $= \cup_{k\geq 0}$DSPACE(2^{n^k})

Planning with Partial Information

Introduction

- Arises when the planning agent doesn't necessarily know the state of the system

Introduction

- Arises when the planning agent doesn't necessarily know the state of the system
- The agent receives some feedback after the execution of an action; it can be
  - Full feedback (the full state is revealed)
  - Partial feedback (e.g. the truth value of a proposition is revealed)
  - Null Feedback

Introduction

- Arises when the planning agent doesn't necessarily know the state of the system
- The agent receives some feedback after the execution of an action; it can be
  - Full feedback (the full state is revealed)
  - Partial feedback (e.g. the truth value of a proposition is revealed)
  - Null Feedback
- After receiving the feedback, the agent chooses the next action

Introduction

- Arises when the planning agent doesn't necessarily know the state of the system
- The agent receives some feedback after the execution of an action; it can be
  - Full feedback (the full state is revealed)
  - Partial feedback (e.g. the truth value of a proposition is revealed)
  - Null Feedback
- After receiving the feedback, the agent chooses the next action
- This is a branch-point in the plan

Introduction

- Arises when the planning agent doesn't necessarily know the state of the system
- The agent receives some feedback after the execution of an action; it can be
  - Full feedback (the full state is revealed)
  - Partial feedback (e.g. the truth value of a proposition is revealed)
  - Null Feedback
- After receiving the feedback, the agent chooses the next action
- This is a branch-point in the plan
- In general, each action depends on the **history** of actions/observations

Formal Model

- Planning problem is tuple $\langle P, A, \Phi_I, \Phi_G, Z \rangle$ where $Z \subseteq P$ is subset of **observable** propositions

Formal Model

- Planning problem is tuple $\langle P, A, \Phi_I, \Phi_G, Z \rangle$ where $Z \subseteq P$ is subset of **observable** propositions

- The agent must consider the subset of possible states at each decision stage (branch)

Formal Model

- Planning problem is tuple $\langle P, A, \Phi_I, \Phi_G, Z\rangle$ where $Z \subseteq P$ is subset of **observable** propositions
- The agent must consider the subset of possible states at each decision stage (branch)
- Subsets of states are known as **belief states**

Formal Model

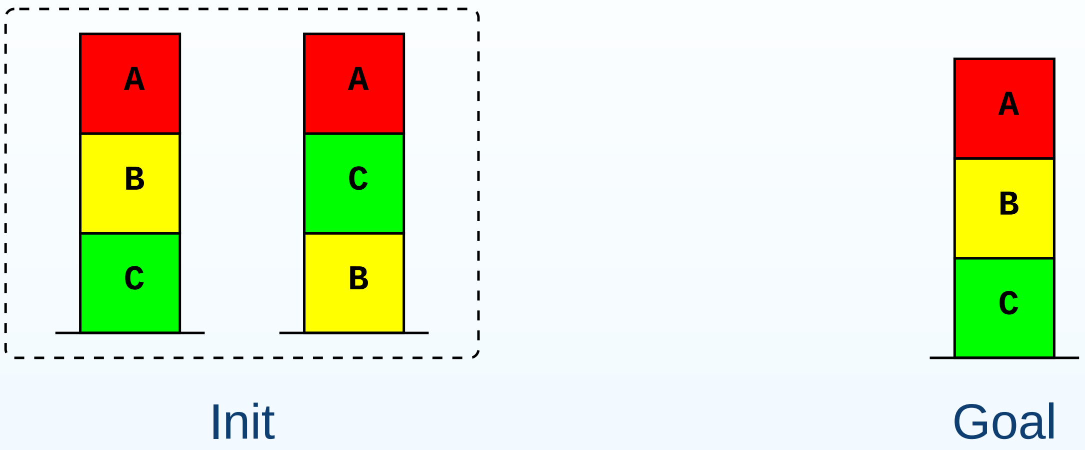
- Planning problem is tuple $\langle P, A, \Phi_I, \Phi_G, Z \rangle$ where $Z \subseteq P$ is subset of **observable** propositions

- The agent must consider the subset of possible states at each decision stage (branch)

- Subsets of states are known as **belief states**

- Plans are tree-like structures that map belief states into actions

Example – Blocksworld (Partial Information)



- Observables: $Z = \{$clear(A), clear(B), clear(C)$\}$
- Contingent Plan:

$$\texttt{pick(A)} \begin{cases} \texttt{stack(A,B)} \\ \texttt{drop(A)}, \texttt{pick(C)}, \texttt{drop(C)}, \texttt{pick(B)}, \texttt{stack(B,C)}, \texttt{pick(A)}, \texttt{stack(A,B)} \end{cases}$$

Complexity of Planning With Partial Information

- Plans are like policies for Partially Observable MDPs (POMDPs)
- Deciding existence of solution for a contingent planning problem with partial observability (i.e. PLAN-PO-CONT) is 2EXPTIME-complete
- 2EXPTIME $= \cup_{k\geq 0}$DTIME$(2^{2^{n^k}})$
- Shown by [Rintanen, 2004] using Alternating TMs with exponential space bound

Another Example – Game of Mastermind

- Played by a codemaker and codebreaker

Another Example – Game of Mastermind

- Played by a codemaker and codebreaker
- The codemaker chooses a secret code, at the beginning of the game, made of 3 colored pegs

Another Example – Game of Mastermind

- Played by a codemaker and codebreaker
- The codemaker chooses a secret code, at the beginning of the game, made of 3 colored pegs
- The task of the codebreaker is to reveal the code by making guesses

Another Example – Game of Mastermind

- Played by a codemaker and codebreaker
- The codemaker chooses a secret code, at the beginning of the game, made of 3 colored pegs
- The task of the codebreaker is to reveal the code by making guesses
- Each guess is a candidate code which is answered by the codebreaker with two tokens of information:
  - the number of exact matches in the guess
  - the number of "near" matches in the guess

Another Example – Game of Mastermind

- Played by a codemaker and codebreaker
- The codemaker chooses a secret code, at the beginning of the game, made of 3 colored pegs
- The task of the codebreaker is to reveal the code by making guesses
- Each guess is a candidate code which is answered by the codebreaker with two tokens of information:
  - the number of exact matches in the guess
  - the number of “near” matches in the guess
- Thus, each guess depends on the information acquired during the game

Another Example – Game of Mastermind

- Played by a codemaker and codebreaker
- The codemaker chooses a secret code, at the beginning of the game, made of 3 colored pegs
- The task of the codebreaker is to reveal the code by making guesses
- Each guess is a candidate code which is answered by the codebreaker with two tokens of information:
  - the number of exact matches in the guess
  - the number of “near” matches in the guess
- Thus, each guess depends on the information acquired during the game
- Although the game can be modeled as a non-deterministic planning problem with partial information, **the goal can’t be represented**

Another Example – Game of Mastermind

- Played by a codemaker and codebreaker
- The codemaker chooses a secret code, at the beginning of the game, made of 3 colored pegs
- The task of the codebreaker is to reveal the code by making guesses
- Each guess is a candidate code which is answered by the codebreaker with two tokens of information:
  - the number of exact matches in the guess
  - the number of “near” matches in the guess
- Thus, each guess depends on the information acquired during the game
- Although the game can be modeled as a non-deterministic planning problem with partial information, **the goal can’t be represented**
- A modal formula is needed to represent such a goal

Modal Formulae

- We use three connectives: $\Box$ (necessity), $\Diamond$ (sufficiency) and $*$ (new one)

Modal Formulae

- We use three connectives: $\Box$ (necessity), $\Diamond$ (sufficiency) and $*$ (new one)
- Semantics defined using triplets (s, b, σ) where s is a state, b is a belief state, and σ is a *sequence* of states:
  - $(s, b, \sigma) \models \Box\varphi$ iff $(s', b, s\sigma) \models \varphi$ for all $s' \in b$,
  - $(s, b, \sigma) \models \Diamond\varphi$ iff $(s', b, s\sigma) \models \varphi$ for some $s' \in b$,
  - $(s, b, s'\sigma) \models \varphi^*$ iff $(s', b, \sigma) \models \varphi$, and
  - $(s, b, \langle\rangle) \not\models \varphi^*$ for all φ.

Modal Formulae

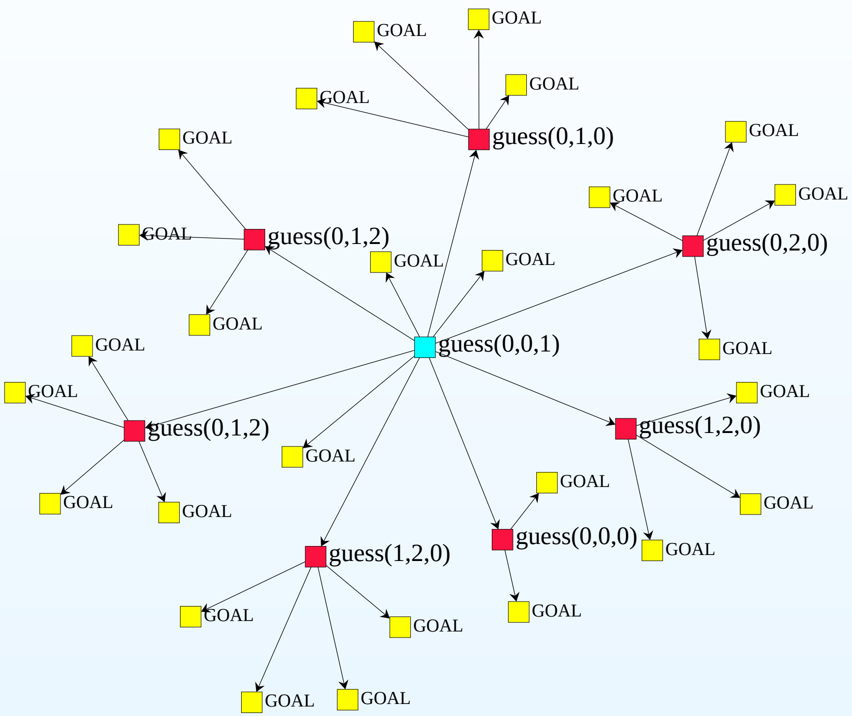
- We use three connectives: $\Box$ (necessity), $\Diamond$ (sufficiency) and $*$ (new one)
- Semantics defined using triplets (s, b, σ) where s is a state, b is a belief state, and σ is a *sequence* of states:
  - $(s, b, \sigma) \models \Box\varphi$ iff $(s', b, s\sigma) \models \varphi$ for all $s' \in b$,
  - $(s, b, \sigma) \models \Diamond\varphi$ iff $(s', b, s\sigma) \models \varphi$ for some $s' \in b$,
  - $(s, b, s'\sigma) \models \varphi^*$ iff $(s', b, \sigma) \models \varphi$, and
  - $(s, b, \langle\rangle) \not\models \varphi^*$ for all φ.
- φ holds in belief b iff $(s, b, \langle\rangle) \models \varphi$ for all $s \in b$

Modal Formulae

- We use three connectives: $\Box$ (necessity), $\Diamond$ (sufficiency) and $*$ (new one)
- Semantics defined using triplets (s, b, σ) where s is a state, b is a belief state, and σ is a *sequence* of states:
  - $(s, b, \sigma) \models \Box\varphi$ iff $(s', b, s\sigma) \models \varphi$ for all $s' \in b$,
  - $(s, b, \sigma) \models \Diamond\varphi$ iff $(s', b, s\sigma) \models \varphi$ for some $s' \in b$,
  - $(s, b, s'\sigma) \models \varphi^*$ iff $(s', b, \sigma) \models \varphi$, and
  - $(s, b, \langle\rangle) \not\models \varphi^*$ for all φ.
- φ holds in belief b iff $(s, b, \langle\rangle) \models \varphi$ for all $s \in b$
- Examples:
  - 'know φ' equivalent to $\Box\varphi \vee \Box\neg\varphi$
  - 'possibly φ' equivalent to $\Diamond\varphi$
  - $\Box\Diamond(p^* \leftrightarrow q)$ true in belief b iff for all $s \in b$ there is $s' \in b$ such that the value of p in s coincides with the value of q in s'

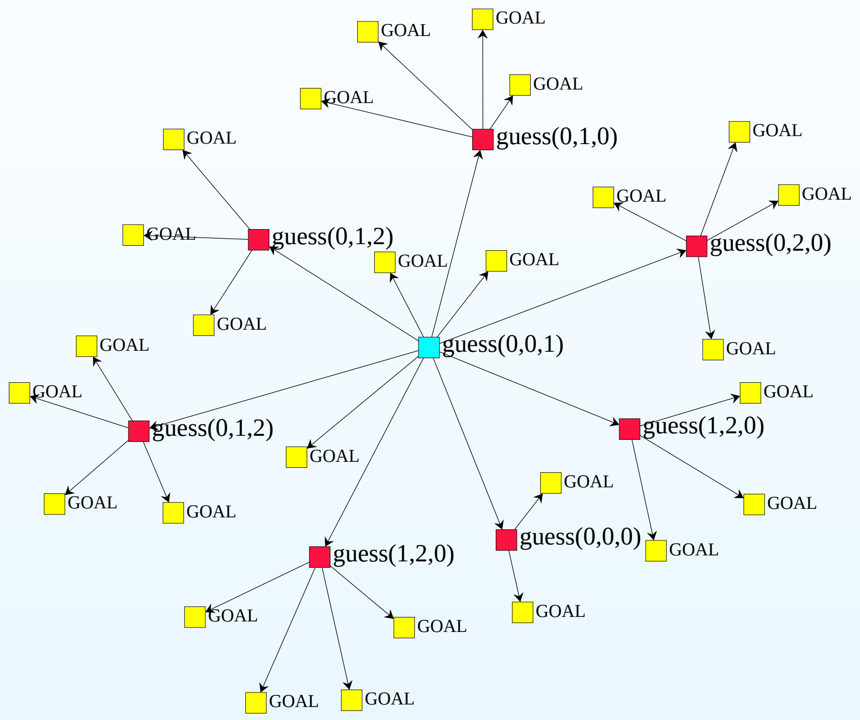
Mastermind Revisited: 3 colors, 3 pegs

- Contingent Plan:



Mastermind Revisited: 3 colors, 3 pegs

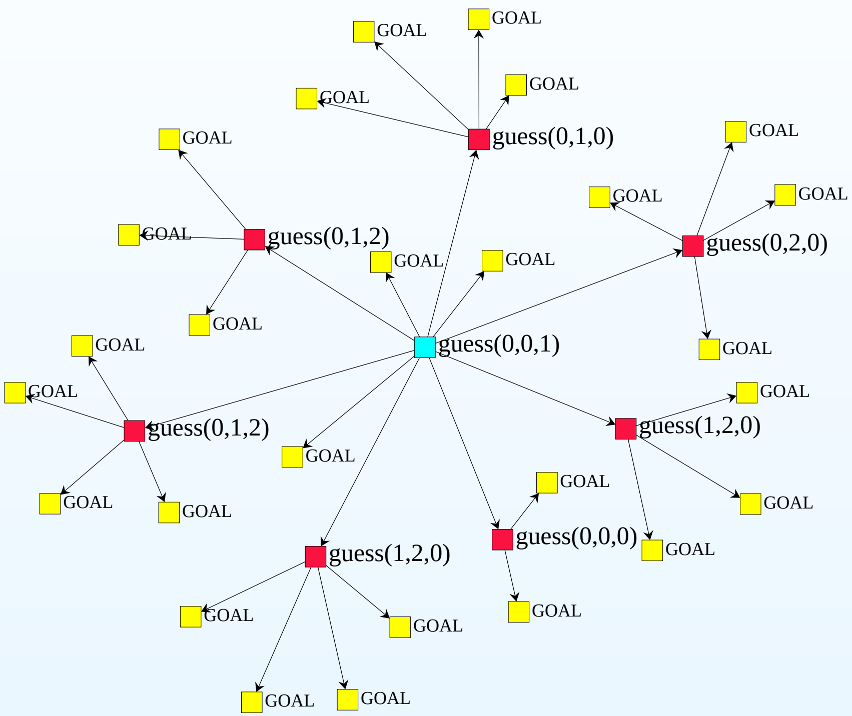
- Contingent Plan:



- As before, we can also compute conformant plans for this task

Mastermind Revisited: 3 colors, 3 pegs

- Contingent Plan:



- As before, we can also compute conformant plans for this task
- The following plan discover the secret code no matter what's its value

$$\text{guess}(2,0,0)\ ,\ \text{guess}(2,1,0)\ ,\ \text{guess}(2,2,1)\,.$$

Complexity of Conformant Planning with Partial Information

- Deciding the existence of solution for a conformant planning problem with partial information (i.e. PLAN-PO-CONF) is 2EXPSPACE-complete

- Deciding the existence of solutions for a contingent planning problem with partial information and with at most k branch points (i.e. PLAN-PO-CONT-k) is also 2EXPSPACE-complete

- If there are no modal formulae in the planning problem, PLAN-PO-CONF and PLAN-PO-CONT-k become EXPSPACE-complete

Summary

Problem	Complete for	Reference
PLAN-PO-CONT	2EXPTIME	[Rintanen, 2004]
PLAN-PO-CONF	2EXPSPACE	**New**
PLAN-PO-CONT-k	2EXPSPACE	**New**

Remember:

- P $\subseteq$ NP $\subseteq$ PSPACE $\subseteq$ EXPTIME $\subseteq$ EXPSPACE $\subseteq$ 2EXPTIME $\subseteq$ 2EXPSPACE

Grand Summary

- New planning tasks considered: solutions of bounded branching and conformant tasks for partially observable problems
- Derived tight bounds on complexity for the new tasks
- Special classes also considered:
  - Problems with partial information and no modal formulae
  - Checking the existence of plans of polynomial length following [Turner, 2002]

References

- T. Bylander. *The Computational Complexity of Propositional STRIPS Planning*. Artificial Intelligence 69: 165–204. 1994.
- R. Goldman & M. Boddy. *Expressive Planning and Explicit Knowledge*. In Proc AIPS'96.
- P. Haslum & P. Jonsson. *Some Results on the Complexity of Planning with Incomplete Information*. In Proc ECP'99, 308–318.
- M. Littman. *Probabilistic Propositional Planning: Representation and Complexity*. In Proc AAAI'97, 748–754.
- J. Rintanen. *Complexity of Planning with Partial Observability*. In Proc ICAPS'04, 345–354.
- D. Smith & D. Weld. *Conformant Graphplan*. In Proc. AAAI'98, 889–896.
- H. Turner. *Polynomial-length Planning Spans the Polynomial Hierarchy*. In Proc JELIA'02, 111–124.