

# Length of plans

Let  $\langle A, I, O, G \rangle$  be a deterministic succinct transition system.

- 1 There is a plan of length 0 iff  $I \models G$ .
- 2 Shortest plans may not be longer than  $2^n - 1$ : If a plan is longer, then it visits some state  $s$  more than once and has the form  $\sigma_1^s \sigma_2^s \sigma_3$ : the plan  $\sigma_1 \sigma_3$  is shorter.
- 3 Shortest plan may have length  $2^n - 1$ : Reach the goal state  $111 \dots 1$  from the initial state  $000 \dots 0$  by an operator that increments the corresponding binary number  $2^n - 1$  times.

AI Planning

Plan length

NP-hardness

Turing machines

Complexity classes

PSPACE-hardness

PSPACE membership

Summary

# Length of plans

Let  $\langle A, I, O, G \rangle$  be a deterministic succinct transition system.

- 1 There is a plan of length 0 iff  $I \models G$ .
- 2 Shortest plans may not be longer than  $2^n - 1$ : If a plan is longer, then it visits some state  $s$  more than once and has the form  $\sigma_1^s \sigma_2^s \sigma_3$ : the plan  $\sigma_1 \sigma_3$  is shorter.
- 3 Shortest plan may have length  $2^n - 1$ : Reach the goal state  $111 \dots 1$  from the initial state  $000 \dots 0$  by an operator that increments the corresponding binary number  $2^n - 1$  times.

AI Planning

Plan length

NP-hardness

Turing machines

Complexity classes

PSPACE-hardness

PSPACE membership

Summary

# Length of plans

Let  $\langle A, I, O, G \rangle$  be a deterministic succinct transition system.

- 1 There is a plan of length 0 iff  $I \models G$ .
- 2 Shortest plans may not be longer than  $2^n - 1$ : If a plan is longer, then it visits some state  $s$  more than once and has the form  $\sigma_1^s \sigma_2^s \sigma_3$ : the plan  $\sigma_1 \sigma_3$  is shorter.
- 3 Shortest plan may have length  $2^n - 1$ : Reach the goal state  $111 \dots 1$  from the initial state  $000 \dots 0$  by an operator that increments the corresponding binary number  $2^n - 1$  times.

AI Planning

Plan length

NP-hardness

Turing machines

Complexity classes

PSPACE-hardness

PSPACE membership

Summary

# Deterministic planning: expressivity

## Definition

The decision problem SAT: test whether a given propositional formula  $\phi$  is satisfiable.

## Reduction from SAT to deterministic planning

$A$  = the set of propositional variables occurring in  $\phi$

$I$  = any state, e.g. all state variables have value 0

$O = (\{\top\} \times A) \cup (\{\{\top, \neg a\} \mid a \in A\})$

There is a plan for  $\langle A, I, O, \phi \rangle$  if and only if  $\phi$  is satisfiable.

AI Planning

Plan length

NP-hardness

Turing machines

Complexity classes

PSPACE-hardness

PSPACE membership

Summary

# Deterministic planning: expressivity

- Because there is a polynomial-time translation from SAT into deterministic planning, and SAT is an NP-complete problem, there is a polynomial time translation from **every decision problem in NP** into deterministic planning. Hence the problem is NP-hard.
- Does deterministic planning have the power of NP, or is it still more powerful?
- We show that it is more powerful: The decision problem of testing whether a plan exists is PSPACE-complete.

AI Planning

Plan length

NP-hardness

Turing machines

Complexity classes

PSPACE-hardness

PSPACE membership

Summary

# Deterministic planning: expressivity

- Because there is a polynomial-time translation from SAT into deterministic planning, and SAT is an NP-complete problem, there is a polynomial time translation from **every decision problem in NP** into deterministic planning. Hence the problem is NP-hard.
- Does deterministic planning have the power of NP, or is it still more powerful?
- We show that it is more powerful: The decision problem of testing whether a plan exists is PSPACE-complete.

AI Planning

Plan length

NP-hardness

Turing machines

Complexity classes

PSPACE-hardness

PSPACE membership

Summary

# Deterministic planning: expressivity

- Because there is a polynomial-time translation from SAT into deterministic planning, and SAT is an NP-complete problem, there is a polynomial time translation from **every decision problem in NP** into deterministic planning. Hence the problem is NP-hard.
- Does deterministic planning have the power of NP, or is it still more powerful?
- We show that it is more powerful: The decision problem of testing whether a plan exists is PSPACE-complete.

AI Planning

Plan length

NP-hardness

Turing machines

Complexity classes

PSPACE-hardness

PSPACE membership

Summary

## Definition

A **Turing machine**  $\langle \Sigma, Q, \delta, q_0, g \rangle$  consists of

- 1 an alphabet  $\Sigma$  (a set of symbols),
- 2 a set  $Q$  of internal states,
- 3 a transition function  $\delta$  that maps  $\langle q, s \rangle$  to a tuple  $\langle s', q', m \rangle$  where  $q, q' \in Q$ ,  $s \in \Sigma \cup \{ |, \square \}$ ,  $s' \in \Sigma$  and  $m \in \{ L, N, R \}$ .
- 4 an initial state  $q_0 \in Q$ , and
- 5 a labeling  $g : Q \rightarrow \{ \text{accept}, \text{reject}, \exists \}$  of states.

# Turing machines

## Example

TM accepting strings  $\epsilon, a, ab, aba, abab, \dots$  is  $\langle \Sigma, Q, \delta, q_1, g \rangle$   
where

$$\Sigma = \{a, b\}$$

$$Q = \{q_1, q_2, q_3, q_4\}$$

$$g(q_1) = \exists$$

$$g(q_3) = \text{accept}$$

$$\delta(q_1, a) = \langle a, q_2, R \rangle$$

$$\delta(q_2, b) = \langle b, q_1, R \rangle$$

$$\delta(q_1, \square) = \langle a, q_3, R \rangle$$

$$\delta(q, s) = \langle a, q_4, R \rangle \text{ for all other } q, s$$

$$g(q_2) = \exists$$

$$g(q_4) = \text{reject}$$

$$\delta(q_1, b) = \langle b, q_4, R \rangle$$

$$\delta(q_2, a) = \langle a, q_4, R \rangle$$

$$\delta(q_2, \square) = \langle b, q_3, R \rangle$$

AI Planning

Plan length

NP-hardness

Turing  
machines

Complexity  
classes

PSPACE-  
hardness

PSPACE  
membership

Summary

# Turing machines

## Example

What does the TM do with the string  $ababb$ ?

$$q_1 \mid \widehat{a}babb \square$$
$$q_2 \mid a\widehat{b}abb \square$$
$$q_1 \mid ab\widehat{a}bb \square$$
$$q_2 \mid aba\widehat{b}b \square$$
$$q_1 \mid ababb\widehat{\phantom{a}} \square$$
$$q_4 \mid ababb\widehat{\phantom{a}} \square$$

The label  $g(q_4) = \text{reject}$ . The TM does not accept the string.

AI Planning

Plan length

NP-hardness

Turing machines

Complexity classes

PSPACE-hardness

PSPACE membership

Summary

# Some complexity classes

## Definition

$DTIME(f)$  is the class of decision problems solved by a **deterministic** Turing machine in  $\mathcal{O}(f(n))$  time when  $n$  is the input string length.  $NTIME(f)$  is defined similarly for **nondeterministic** Turing machines.

$DSPACE(f)$  is the class of decision problems solved by a **deterministic** Turing machine in  $\mathcal{O}(f(n))$  space when  $n$  is the input string length.

$$EXSPACE = \bigcup_{k \geq 0} DSPACE(2^{n^k})$$

$$NEXPTIME = \bigcup_{k \geq 0} NTIME(2^{n^k})$$

$$EXPTIME = \bigcup_{k \geq 0} DTIME(2^{n^k})$$

$$PSPACE = \bigcup_{k \geq 0} DSPACE(n^k)$$

$$NP = \bigcup_{k \geq 0} NTIME(n^k)$$

$$P = \bigcup_{k \geq 0} DTIME(n^k)$$

AI Planning

Plan length

NP-hardness

Turing machines

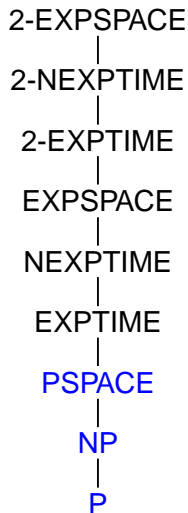
Complexity classes

PSPACE-hardness

PSPACE membership

Summary

# Some complexity classes



AI Planning

Plan length

NP-hardness

Turing machines

Complexity classes

PSPACE-hardness

PSPACE membership

Summary

# Simulation of PSPACE Turing machines

Close match between space-bounded Turing machines and planning problems.

- 1 Turing machine configurations  $\sim$  states
- 2 Turing machine transitions  $\sim$  operators
- 3 initial configuration  $\sim$  initial state
- 4 accepting configurations  $\sim$  goal states

For simulation of PSPACE TMs a number of state variables and operators that is **polynomial** in input string length suffices.

AI Planning

Plan length

NP-hardness

Turing machines

Complexity classes

PSPACE-hardness

Example

State variables

Initial state

Goal formula

Operators

Example

PSPACE membership

Summary

# Simulation of PSPACE Turing machines

We show how polynomial-space Turing machines can be simulated by planning.

- Tape cell contents are represented in state variables.
- R/W head location is represented in state variables.
- Internal TM state is represented in state variables.
- Transitions are represented as operators.

## Theorem

*A Turing machine  $M$  accepts an input string  $\sigma$  if and only if  $T(M, \sigma) = \langle A, I, O, G \rangle$  has a plan.*

AI Planning

Plan length

NP-hardness

Turing machines

Complexity classes

PSPACE-hardness

Example

State variables

Initial state

Goal formula

Operators

Example

PSPACE membership

Summary

# Simulation of PSPACE Turing machines

Turing machine with  $\Sigma = \{a, b, c\}$ , input string of length  $n = 4$ , space bound  $p(n) = n^2 = 16$ , internal states  $Q = \{q_1, q_2, q_3\}$ .

State variables in the corresponding planning problem:

tape cell:	state variables for tape cells						
	0	1	2	3	...	15	16
R/W head:	$h_0$	$h_1$	$h_2$	$h_3$	...	$h_{15}$	$h_{16}$
symbol $a$ :	$a_0$	$a_1$	$a_2$	$a_3$	...	$a_{15}$	$a_{16}$
symbol $b$ :	$b_0$	$b_1$	$b_2$	$b_3$	...	$b_{15}$	$b_{16}$
symbol $c$ :	$c_0$	$c_1$	$c_2$	$c_3$	...	$c_{15}$	$c_{16}$
symbol $\square$ :	$\square_0$	$\square_1$	$\square_2$	$\square_3$	...	$\square_{15}$	$\square_{16}$
symbol $ $ :	$ _0$	$ _1$	$ _2$	$ _3$	...	$ _{15}$	$ _{16}$
state $q_1$ :	$q_1$						
state $q_2$ :	$q_2$						
state $q_3$ :	$q_3$						

AI Planning

Plan length

NP-hardness

Turing machines

Complexity classes

PSPACE-hardness

Example  
State variables  
Initial state  
Goal formula  
Operators  
Example

PSPACE membership

Summary

# Simulation of PSPACE Turing machines

## Example

TM config.	state variable values					plan	
	1	2	3	4	5		
$q_1   \widehat{a}babbb \square$	$\widehat{1}00$	$0\widehat{1}0$	$100$	$0\widehat{1}0$	$0\widehat{1}0$	$1000$	$o_{a,q_1,1}$
$q_2   \widehat{b}babbb \square$	$0\widehat{1}0$	$\widehat{0}10$	$100$	$0\widehat{1}0$	$0\widehat{1}0$	$0\widehat{1}00$	$o_{b,q_2,2}$
$q_4   bc\widehat{a}bb \square$	$0\widehat{1}0$	$00\widehat{1}$	$\widehat{1}00$	$0\widehat{1}0$	$0\widehat{1}0$	$000\widehat{1}$	$o_{a,q_4,3}$
$q_3   bcb\widehat{b}b \square$	$0\widehat{1}0$	$00\widehat{1}$	$0\widehat{1}0$	$\widehat{0}10$	$0\widehat{1}0$	$00\widehat{1}0$	$o_{b,q_3,4}$
$q_1   bcb\widehat{c}cb \square$	$0\widehat{1}0$	$00\widehat{1}$	$\widehat{0}10$	$00\widehat{1}$	$0\widehat{1}0$	$1000$	$o_{b,q_1,3}$
$q_4   bc\widehat{a}acb \square$	$0\widehat{1}0$	$\widehat{0}0\widehat{1}$	$100$	$00\widehat{1}$	$0\widehat{1}0$	$000\widehat{1}$	$o_{c,q_4,2}$
$q_4   \widehat{b}aacb \square$	$\widehat{0}10$	$100$	$100$	$00\widehat{1}$	$0\widehat{1}0$	$000\widehat{1}$	$o_{b,q_4,1}$

Operator  $o_{s,q,h}$  is applicable when current symbol is  $s$ , current TM state is  $q$ , and current tape cell is  $h$ .

AI Planning

Plan length

NP-hardness

Turing machines

Complexity classes

PSPACE-hardness

Example

State variables

Initial state

Goal formula

Operators

Example

PSPACE membership

Summary

# PSPACE simulation

Simulate a TM  $\langle \Sigma, Q, \delta, q_0, g \rangle$  that needs at most  $p(n)$  tape cells on an input string of length  $n$ .

State variables in the succinct transition system are

- 1  $\{q_1, \dots, q_{|Q|}\} = Q$  for the current state of the TM,
- 2  $s_i$  for every symbol  $s \in \Sigma \cup \{|\, \square\}$  and tape cell  $i \in \{0, \dots, p(n)\}$ ,
- 3  $h_i$  for every  $i \in \{0, \dots, p(n)\}$  (position of the R/W head).

AI Planning

Plan length

NP-hardness

Turing machines

Complexity classes

PSPACE-hardness

Example

State variables

Initial state

Goal formula

Operators

Example

PSPACE membership

Summary

# PSPACE simulation

Initial state

- 1  $I(q_0) = 1$  and  $I(q) = 0$  for all  $q \in Q \setminus \{q_0\}$ .
- 2  $I(s_i) = 1$  if  $i \leq n$  and input symbol  $i$  is  $s$ .
- 3  $I(s_i) = 0$  if  $i \leq n$  and  $s \in S$  and input symbol  $i$  is not  $s$ .
- 4  $I(\square_i) = 1$  iff  $i \in \{n + 1, \dots, p(n) - 1\}$
- 5  $I(|_i) = 1$  iff  $i = 0$
- 6  $I(h_i) = 1$  iff  $i = 1$

AI Planning

Plan length

NP-hardness

Turing machines

Complexity classes

PSPACE-hardness

Example

State variables

Initial state

Goal formula

Operators

Example

PSPACE membership

Summary

# PSPACE simulation

Goal formula

Goal formula requires that the Turing machine is in an accepting state.

$$G = \bigvee \{q \in Q \mid g(q) = \text{accept}\}.$$

AI Planning

Plan length

NP-hardness

Turing machines

Complexity classes

PSPACE-hardness

Example

State variables

Initial state

**Goal formula**

Operators

Example

PSPACE membership

Summary

# PSPACE simulation

## Operators

For all  $s \in \Sigma \cup \{ \sqcup, \square \}$  and  $q \in Q$  and  $i \in \{0, \dots, p(n)\}$  with  $\delta(q, s) = \langle s', q', m \rangle$  such that  $m \neq R$  or  $i < p(n)$  define

$$o_{s,q,i} = \langle h_i \wedge s_i \wedge q, \nu \wedge \chi \wedge \mu \rangle$$

where

$\nu$  describes the writing operation,

$\chi$  describes the change in the internal state of the TM,

$\mu$  describes the movement of the R/W head.

The requirement  $m \neq R$  or  $i < p(n)$  means that no transition violating the space bound is possible.

Operator  $o_{s,q,i}$  corresponds to the **unique transition** from a configuration where current symbol is  $s$ , internal state is  $q$ , and R/W head location is  $i$ .

AI Planning

Plan length

NP-hardness

Turing machines

Complexity classes

PSPACE-hardness

Example

State variables

Initial state

Goal formula

Operators

Example

PSPACE membership

Summary

# PSPACE simulation

## Operators' effects

symbol written onto the tape

$$\nu = \begin{cases} \top & \text{if } s \in \{|\, s'\} \\ \neg s_i \wedge s'_i & \text{otherwise} \end{cases}$$

change in the internal state

$$\chi = \begin{cases} \neg q \wedge q' & \text{if } q \neq q' \\ \top & \text{otherwise} \end{cases}$$

movement of the R/W head

$$\mu = \begin{cases} \neg h_i \wedge h_{i-1} & \text{if } i > 0 \text{ and } m = L \\ \neg h_i \wedge h_{i+1} & \text{if } i < p(n) \text{ and } m = R \\ \top & \text{otherwise} \end{cases}$$

AI Planning

Plan length

NP-hardness

Turing machines

Complexity classes

PSPACE-hardness

Example

State variables

Initial state

Goal formula

Operators

Example

PSPACE membership

Summary

# PSPACE simulation

## Example

- 1 Turing machine  $\langle \{a, b\}, \{q_1, q_2, q_{acc}\}, \delta, q_1, g \rangle$  where  $\delta$  is

	$a$	$b$	$ $	$\square$
$q_1$	$\langle a, q_1, R \rangle$	$\langle b, q_2, N \rangle$	$\langle  , q_2, R \rangle$	$\langle b, q_1, N \rangle$
$q_2$	$\langle a, q_1, L \rangle$	$\langle a, q_{acc}, N \rangle$	$\langle  , q_1, R \rangle$	$\langle a, q_2, L \rangle$
$q_{acc}$	—	—	—	—

and  $g(q_{acc}) = \text{accept}$ ,  $g(q_1) = \exists$  and  $g(q_2) = \exists$ .

(This Turing machine does not do anything interesting!)

- 2 Input string is abaab.
- 3 Let the space bound be  $p(5) = 7$  for some polynomial  $p$ .

# PSPACE simulation

## Example

The succinct transition system corresponding to the Turing machine is  $\langle A, I, O, G \rangle$  where

- 1  $A = \{q_1, q_2, q_{acc}, h_0, \dots, h_7, a_0, \dots, a_7, b_0, \dots, b_7, \dots\}$ ,
- 2  $I \models h_1 \wedge |_0 \wedge a_1 \wedge b_2 \wedge a_3 \wedge a_4 \wedge b_5 \wedge \square_6 \wedge \square_7 \wedge \neg h_0 \wedge \neg a_0 \wedge \neg b_0 \wedge \neg \square_0 \wedge \dots$ ,
- 3 operators in  $O$  are on the next slide, and
- 4  $G = q_{acc}$ .

AI Planning

Plan length

NP-hardness

Turing machines

Complexity classes

PSPACE-hardness

Example

State variables

Initial state

Goal formula

Operators

Example

PSPACE membership

Summary

# PSPACE simulation

## Example

Only part of the about

$|\{0, 1, \dots, 7\}| \times |\{q_1, q_2\}| \times |\{a, b, \sqcup, \square\}|$  operators are given below, for R/W head position 1 and input symbols a and b:

$$O = \{ \langle h_1 \wedge a_1 \wedge q_1, \neg h_1 \wedge h_2 \rangle, \dots, \\ \langle h_1 \wedge b_1 \wedge q_1, \neg q_1 \wedge q_2 \rangle, \dots, \\ \langle h_1 \wedge a_1 \wedge q_2, \neg q_2 \wedge q_1 \wedge \neg h_1 \wedge h_0 \rangle, \dots, \\ \langle h_1 \wedge b_1 \wedge q_2, \neg b_1 \wedge a_1 \wedge \neg q_2 \wedge q_{acc} \rangle, \dots \}$$

AI Planning

Plan length

NP-hardness

Turing machines

Complexity classes

PSPACE-hardness

Example

State variables

Initial state

Goal formula

Operators

Example

PSPACE membership

Summary

# Deterministic planning is solvable in PSPACE

- The PSPACE-hardness result provides a **lower bound** on the complexity of deterministic planning. Is the problem hard for a complexity class more difficult than PSPACE?
- We next give an **upper bound** on the complexity by showing that the problem belongs to PSPACE.
- Hence the problem is **PSPACE-complete**, locating the problem exactly in one complexity class.
- It is not known whether  $NP \neq PSPACE$  or even  $P \neq PSPACE$ , but the result is still useful because for all practical purposes we can assume that  $NP \neq PSPACE$ .
- For example, we may conclude that **there is no polynomial-time translation from planning to the satisfiability problem** (the translation we gave earlier is **linear in the plan length**, which may be **exponential in the problem instance size**).

AI Planning

Plan length

NP-hardness

Turing machines

Complexity classes

PSPACE-hardness

PSPACE membership

Idea

Algorithm

Properties

Summary

# Deterministic planning is solvable in PSPACE

- The PSPACE-hardness result provides a **lower bound** on the complexity of deterministic planning. Is the problem hard for a complexity class more difficult than PSPACE?
- We next give an **upper bound** on the complexity by showing that the problem belongs to PSPACE.
- Hence the problem is **PSPACE-complete**, locating the problem exactly in one complexity class.
- It is not known whether  $NP \neq PSPACE$  or even  $P \neq PSPACE$ , but the result is still useful because for all practical purposes we can assume that  $NP \neq PSPACE$ .
- For example, we may conclude that **there is no polynomial-time translation from planning to the satisfiability problem** (the translation we gave earlier is linear in the plan length, which may be exponential in the problem instance size.).

AI Planning

Plan length

NP-hardness

Turing machines

Complexity classes

PSPACE-hardness

PSPACE membership

Idea

Algorithm

Properties

Summary

# Deterministic planning is solvable in PSPACE

- The PSPACE-hardness result provides a **lower bound** on the complexity of deterministic planning. Is the problem hard for a complexity class more difficult than PSPACE?
- We next give an **upper bound** on the complexity by showing that the problem belongs to PSPACE.
- Hence the problem is **PSPACE-complete**, locating the problem exactly in one complexity class.
- It is not known whether  $NP \neq PSPACE$  or even  $P \neq PSPACE$ , but the result is still useful because for all practical purposes we can assume that  $NP \neq PSPACE$ .
- For example, we may conclude that **there is no polynomial-time translation from planning to the satisfiability problem** (the translation we gave earlier is linear in the plan length, which may be exponential in the problem instance size.).

AI Planning

Plan length

NP-hardness

Turing machines

Complexity classes

PSPACE-hardness

PSPACE membership

Idea

Algorithm

Properties

Summary

# Deterministic planning is solvable in PSPACE

- The PSPACE-hardness result provides a **lower bound** on the complexity of deterministic planning. Is the problem hard for a complexity class more difficult than PSPACE?
- We next give an **upper bound** on the complexity by showing that the problem belongs to PSPACE.
- Hence the problem is **PSPACE-complete**, locating the problem exactly in one complexity class.
- It is not known whether  $NP \neq PSPACE$  or even  $P \neq PSPACE$ , but the result is still useful because for all practical purposes we can assume that  $NP \neq PSPACE$ .
- For example, we may conclude that **there is no polynomial-time translation from planning to the satisfiability problem** (the translation we gave earlier is **linear in the plan length**, which may be **exponential in the problem instance size**).

AI Planning

Plan length

NP-hardness

Turing machines

Complexity classes

PSPACE-hardness

PSPACE membership

Idea

Algorithm

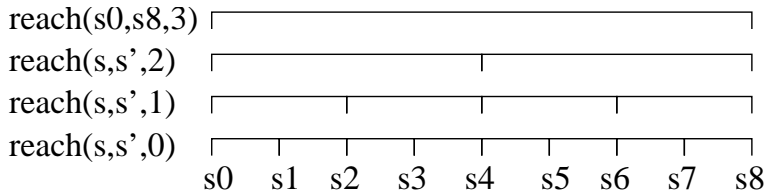
Properties

Summary

# Deterministic planning is solvable in PSPACE

## Proof idea

Recursive algorithm for testing  $m$ -step reachability between two states with  $\log m$  memory consumption.



AI Planning

Plan length

NP-hardness

Turing machines

Complexity classes

PSPACE-hardness

PSPACE membership

Idea

Algorithm

Properties

Summary

# Deterministic planning is solvable in PSPACE

## Algorithm

Testing whether a plan of length  $\leq 2^n$  exists:

```
PROCEDURE reach( $s, s', n$ )  
IF  $n = 0$  THEN  
    IF  $s = s'$  OR  $s' = \text{app}_o(s)$  for some  $o \in O$   
    THEN RETURN true  
    ELSE RETURN false;  
ELSE  
    FOR all states  $s''$  DO  
        IF reach( $s, s'', n - 1$ ) AND reach( $s'', s', n - 1$ )  
        THEN RETURN true  
    END  
    RETURN false;
```

This algorithm does not store the plan anywhere (it could not without violating the space bound!) but could be modified to output it.

AI Planning

Plan length

NP-hardness

Turing machines

Complexity classes

PSPACE-hardness

PSPACE membership

Idea

Algorithm

Properties

Summary

# Deterministic planning is solvable in PSPACE

Correctness of the algorithm

## Correctness

For a succinct transition system  $N$  with  $n$  state variables,  $N$  has a plan if and only if  $\text{reach}(I, s', n)$  returns true for some  $s'$  such that  $s' \models G$ .

## Memory consumption

If number of states is  $2^n$ , then recursion depth is  $n$ . At each recursive call only one state  $s''$  is represented, taking space  $\mathcal{O}(n)$ , which means that total memory consumption at any time point is  $\mathcal{O}(n^2)$ , which is polynomial in the size of the succinct transition system.

AI Planning

Plan length

NP-hardness

Turing machines

Complexity classes

PSPACE-hardness

PSPACE membership

Idea

Algorithm

Properties

Summary

# Summary

- For  $n$  Boolean state variables shortest plans have length  $\leq 2^n - 1$ .
- Testing for the existence of a plan is PSPACE-hard: The halting problem of every deterministic polynomial-space Turing machine can be translated into a deterministic planning problem.
- Testing for the existence of a plan can be done in PSPACE.

AI Planning

Plan length

NP-hardness

Turing machines

Complexity classes

PSPACE-hardness

PSPACE membership

Summary

# Summary

- For  $n$  Boolean state variables shortest plans have length  $\leq 2^n - 1$ .
- Testing for the existence of a plan is PSPACE-hard: The halting problem of every deterministic polynomial-space Turing machine can be translated into a deterministic planning problem.
- Testing for the existence of a plan can be done in PSPACE.

AI Planning

Plan length

NP-hardness

Turing machines

Complexity classes

PSPACE-hardness

PSPACE membership

Summary

# Summary

- For  $n$  Boolean state variables shortest plans have length  $\leq 2^n - 1$ .
- Testing for the existence of a plan is PSPACE-hard: The halting problem of every deterministic polynomial-space Turing machine can be translated into a deterministic planning problem.
- Testing for the existence of a plan can be done in PSPACE.

AI Planning

Plan length

NP-hardness

Turing machines

Complexity classes

PSPACE-hardness

PSPACE membership

Summary