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 $111\ldots1$ from the initial state $000\ldots0$ by an operator that increments the corresponding binary number $2^n - 1$ times.

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.

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 $111\ldots1$ from the initial state $000\ldots0$ by an operator that increments the corresponding binary number $2^n - 1$ times.

Deterministic planning: expressivity

Definition

The decision problem SAT: test whether a given propositional formula ϕ is satisfiable.

Reduction from SAT to deterministic planning

$$A = \text{the set of propositional variables occurring in } \phi$$
$$I = \text{any state, e.g. all state variables have value 0}$$
$$O = (\{\top\} \times A) \cup (\{\langle \top, \neg a \rangle | a \in A\})$$

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

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.

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.

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.

Turing machines

Definition

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

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

Turing machines
Example

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

$$\Sigma = \{a, b\}$$
$$Q = \{q_1, q_2, q_3, q_4\}$$
$$g(q_1) = \exists \qquad g(q_2) = \exists$$
$$g(q_3) = \text{accept} \qquad g(q_4) = \text{reject}$$
$$\delta(q_1, a) = \langle a, q_2, R \rangle \qquad \delta(q_1, b) = \langle b, q_4, R \rangle$$
$$\delta(q_2, b) = \langle b, q_1, R \rangle \qquad \delta(q_2, a) = \langle a, q_4, R \rangle$$
$$\delta(q_1, \Box) = \langle a, q_3, R \rangle \qquad \delta(q_2, \Box) = \langle b, q_3, R \rangle$$
$$\delta(q, s) = \langle a, q_4, R \rangle \text{ for all other } q, s$$

What does the TM do with the string ababb?

$$
\begin{array}{l|l}
q_1 & \widehat{a}babb\Box \\
q_2 & a\widehat{b}abb\Box \\
q_1 & ab\widehat{a}bb\Box \\
q_2 & aba\widehat{b}b\Box \\
q_1 & abab\widehat{b}\Box \\
q_4 & ababb\widehat{\Box}
\end{array}
$$

The label $g(q_4) =$ *reject*. The TM does not accept the string.

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.

$$
\begin{aligned}
\text{EXPSPACE} &= \bigcup_{k\geq 0} \text{DSPACE}(2^{n^k}) \\
\text{NEXPTIME} &= \bigcup_{k\geq 0} \text{NTIME}(2^{n^k}) \\
\text{EXPTIME} &= \bigcup_{k\geq 0} \text{DTIME}(2^{n^k}) \\
\text{PSPACE} &= \bigcup_{k\geq 0} \text{DSPACE}(n^k) \\
\text{NP} &= \bigcup_{k\geq 0} \text{NTIME}(n^k) \\
\text{P} &= \bigcup_{k\geq 0} \text{DTIME}(n^k)
\end{aligned}
$$

Some complexity classes

2-EXPSPACE
|
2-NEXPTIME
|
2-EXPTIME
|
EXPSPACE
|
NEXPTIME
|
EXPTIME
|
PSPACE
|
NP
|
P

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.

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 σ if and only if $T(M, \sigma) = \langle A, I, O, G \rangle$ has a plan.

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:

	state variables for tape cells						
tape cell:	0	1	2	3	$\cdots$	15	16
R/W head:	h_0	h_1	h_2	h_3	$\cdots$	h_{15}	h_{16}
symbol a:	a_0	a_1	a_2	a_3	$\cdots$	a_{15}	a_{16}
symbol b:	b_0	b_1	b_2	b_3	$\cdots$	b_{15}	b_{16}
symbol c:	c_0	c_1	c_2	c_3	$\cdots$	c_{15}	c_{16}
symbol $\Box$:	$\Box_0$	$\Box_1$	$\Box_2$	$\Box_3$	$\cdots$	$\Box_{15}$	$\Box_{16}$
symbol $\vert$:	$\vert_0$	$\vert_1$	$\vert_2$	$\vert_3$	$\cdots$	$\vert_{15}$	$\vert_{16}$
state q_1:	q_1						
state q_2:	q_2						
state q_3:	q_3						

Simulation of PSPACE Turing machines

Example

	state variable values						
	1	2	3	4	5		
TM config.	abc	abc	abc	abc	abc	$q_1q_2q_3q_4$	plan
$q_1 \mid \hat{a}babb\square$	$\widehat{100}$	010	100	010	010	1000	$o_{a,q_1,1}$
$q_2 \mid b\hat{b}abb\square$	010	$\widehat{010}$	100	010	010	0100	$o_{b,q_2,2}$
$q_4 \mid bc\hat{a}bb\square$	010	001	$\widehat{100}$	010	010	0001	$o_{a,q_4,3}$
$q_3 \mid bcb\hat{b}b\square$	010	001	010	$\widehat{010}$	010	0010	$o_{b,q_3,4}$
$q_1 \mid bc\hat{b}cb\square$	010	001	$\widehat{010}$	001	010	1000	$o_{b,q_1,3}$
$q_4 \mid b\hat{c}acb\square$	010	$\widehat{001}$	100	001	010	0001	$o_{c,q_4,2}$
$q_4 \mid \hat{b}aacb\square$	$\widehat{010}$	100	100	001	010	0001	$o_{b,q_4,1}$

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

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

PSPACE simulation
Initial state

1. $I(q_0) = 1$ and $I(q) = 0$ for all $q \in Q \backslash \{q_0\}$.
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. $I(\Box_i) = 1$ iff $i \in \{n+1, \ldots, p(n) - 1\}$
5. $I(|_i) = 1$ iff $i = 0$
6. $I(h_i) = 1$ iff $i = 1$

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

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

PSPACE simulation
Operators

For all $s \in \Sigma \cup \{|, \Box\}$ and $q \in Q$ and $i \in \{0, \ldots, p(n)\}$ 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

ν describes the writing operation,

χ describes the change in the internal state of the TM,

μ 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.

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

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

PSPACE simulation
Example

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

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

   and $g(q_{acc}) =$ 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.

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

1. $A = \{q_1, q_2, q_{acc}, h_0, \ldots, h_7, a_0, \ldots, a_7, b_0, \ldots b_7, \ldots\}$,
2. $I \models h_1 \wedge |_0 \wedge a_1 \wedge b_2 \wedge a_3 \wedge a_4 \wedge b_5 \wedge \Box_6 \wedge \Box_7 \wedge \neg h_0 \wedge \neg a_0 \wedge \neg b_0 \wedge \neg \Box_0 \wedge \cdots$,
3. operators in O are on the next slide, and
4. $G = q_{acc}$.

Only part of the about $|\{0,1,\dots,7\}| \times |\{q_1,q_2\}| \times |\{a,b,|,\Box\}|$ operators are given below, for R/W head position 1 and input symbols a and b:

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

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≠PSPACE or even P≠PSPACE, but the result is still useful because for all practical purposes we can assume that NP≠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.).

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.).

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.).

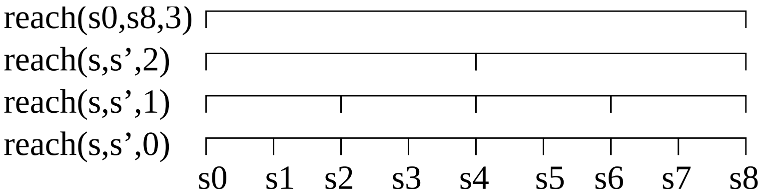
Deterministic planning is solvable in PSPACE

- 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.).

Deterministic planning is solvable in PSPACE

Proof idea

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



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 $s' = app_o(s)$ 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.

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 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.

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.

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.

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.