

Computational Complexity of Plan and Controller Synthesis under Partial Observability (not quite finished)

Jussi Rintanen
Albert-Ludwigs-Universität Freiburg, Germany

We carry out an analysis on the computational properties of a number of controller/plan synthesis problems as investigated in AI planning, discrete-event systems, and related areas. Our focus of interest is synthesis problems with partial observability, that is, the controllers have feedback/observations, but the observations do not in general allow determining the current state of the system unambiguously. The controller-synthesis problems arising in this setting have a very high complexity, yielding problems complete for the complexity class 2-EXP, one of few natural ones that are known. In addition to the general problem, we investigate the complexity of the synthesis problem under several natural restrictions, including the cases when the transitions are deterministic, that is, the changes caused by a given transition in a given state are always the same, and deterministic state-independent transitions, that is, the changes caused by a given transition are always the same independently of the state.

Contents

1	Introduction	2
2	Preliminaries: Computational complexity	4
3	Preliminaries: Problem Definition	6
3.1	Transition systems	6
3.2	Succinct transition systems	7
3.3	Alternative observation models	9
3.4	Plans	10
3.5	Decision problems	11
3.6	Reductions between decision problems	13
4	Complexity of Planning for Reachability: Lower Bounds	14
4.1	Planning without observability	19
4.2	Planning with partial observability	22
4.3	Plans with loops	25
5	Complexity of Planning for Maintenance	28
6	Complexity of Planning for Repeated Reachability: Upper Bounds	29
6.1	Planning with full observability	29
6.2	Planning with unobservability	34
6.3	Planning with partial observability	37
7	Summary of the Results	42
7.1	Lower bounds for reachability	42
7.2	Upper bounds for repeated reachability	42

8 Implications to Discrete Event Systems **43**

9 Related Work **44**

9.1 Controller synthesis for discrete event systems 44

9.2 Planning 44

9.3 MDPs and probabilistic planning 45

9.4 Game theory 46

9.5 Temporal logic program synthesis 47

10 Conclusions **47**

1. INTRODUCTION

Planning and different types of controller synthesis address an important problem faced in decision making in a dynamic changing environment: what actions to take to achieve given goals. These problems have been addressed in the framework of Markov decision processes (MDPs) [Howard 1960; Puterman 1994] where the focus has been the construction of policies, courses of actions, that produce the highest possible or almost highest possible rewards.

Acting in complex environments usually involves partial observability. This means that the current state of the world cannot be observed exactly, and taking best possible actions becomes more difficult. The MDP framework has been extended to partially observability [Smallwood and Sondik 1973; Kaelbling, Littman, and Cassandra 1998].

However, partial observability together with the optimality requirement leads to computational difficulties. Madani et al. [2003] have shown that the most general plan and controller synthesis problems are unsolvable when optimality or utility exceeding a given value is required in the presence of partial observability. For time horizons of a bounded length the plan and controller synthesis problems are solvable but have a high complexity [Mundhenk, Goldsmith, Lusena, and Allender 2000].

Instead of a priori imposing upper bounds on horizon lengths, there are other ways of tackling the difficulties posed by partial observability. The same plan and controller synthesis problems can be addressed also without exact information on the transition probabilities of nondeterministic actions. In addition to making the problems slightly easier, there may be other reasons for not considering probabilities, including the difficulty of obtaining accurate probabilities and because of the interpretation of nondeterminism as a result of an adversarial environment that tries to prevent the agent from achieving the goals.

There are two research areas that have considered non-probabilistic plan and controller synthesis problems. The first is planning in Artificial Intelligence, where the problem of conditional planning can be viewed as a non-probabilistic version of the POMDP problem [Weld, Anderson, and Smith 1998; Bonet and Geffner 2000; Bertoli, Cimatti, Roveri, and Traverso 2001; Rintanen 2002]. When probabilities are ignored, or equivalently, it is required that the goals are achieved with probability 1, the plan and controller synthesis problems become solvable. These synthesis problems are still very interesting, as in many applications meeting the objectives with probability 1 is possible, or the environment may be modified to make it possible, or the failure probabilities are so close to 0 that they may be ignored.

Along with AI planning, closely related controller synthesis problems have been investigated in connection with Discrete event systems (DES) [Ramadge and Wonham 1987b;

Ramadge and Wonham 1987a]. Like AI planning, also DES have been investigated in connection of factored state variable based representations of transition systems. Two types of factored representations that have been a focus of interest are based on Petri nets [Petri 1962; Murata 1989; Reisig 1985] and vector-addition systems (VASS) [Hopcroft and Pansiot 1979]. Works on DES using Petri nets include [Denham 1988; Ichikawa and Hiraishi 1988; Sreenivas and Krogh 1991], and those using VASS include [Li and Wonham 1993].

These models differ from the ones used in this work as we consider only state variables with a finite number of possible values, whereas Petri nets and VASS in general do not have this restriction. Classes of Petri nets and VASS of high practical interest are those having this kind of finite bound. For Petri nets the existence of an upper bound n on the number of tokens in a node is called n -safety. Petri nets that are n -safe have only a finite number of states.

Another difference to the DES controller synthesis problem is the notion of control that is used. A DES model consists of a set of transitions, a subset of which – the controllable transitions – can be disabled at any moment of time. Any transition that is enabled (not disabled) can fire. The controller synthesis problem is to choose for all states (or belief states when observability is partial) a set of enabled transitions to satisfy the objective the controller tries to achieve. DES can be embedded in our framework as discussed in Section 8.

Like the AI planning problem, also the DES controller synthesis problem has been considered in the setting of partial observability [Lin and Wonham 1988; Cho and Marcus 1989; Inan 1994; Kumar and Shayman 1997; Marchand, Boivineau, and Lafortune 2001].

In this work we analyze the computational complexity of (non-probabilistic) plan and controller synthesis problems, especially in the presence of restrictions on observability. This is one of the central problems in artificial intelligence, the decision making of intelligent agents in complex environments, as well as many related areas, ranging from (simple forms of) program synthesis to intelligent manufacturing systems.

Our goal in this work is to identify the algorithmic basis of plan/controller synthesis for transition systems represented succinctly in terms of state variables. For each problem variant we establish the membership of the problem in a complexity class by giving an (abstract) algorithm for the problem, and prove that the problem is hard for the complexity class in question, showing that the algorithm is optimal. This approach, the use of the complexity theoretic notion of completeness, provides a solid foundation for the development of more concrete and efficient algorithms for the problems in question. The work covers a class of discrete non-probabilistic planning and controller synthesis problems, strengthening the formal foundation for the development of efficient synthesis algorithms.

The initial motivation for investigating the complexity of succinct representations of computational problems was that, possibly, the subset of problem instances representable succinctly (in our case, having a representation that in many cases has only a logarithmic size in the cardinality of the state space) might be easier to solve than the general problem [Papadimitriou and Yannakakis 1986; Lozano and Balcázar 1990]. However, this turned out not to be the case for many graph-theoretic problems: exponential decrease in the instance size causes a corresponding increase in the worst-case complexity, and hence the worst-case complexity in terms of non-succinct instance size is the same.

This is also, not surprisingly, the case in some of the more general problems analyzed in this work. Succinct representations may increase problem complexity exponentially, and partial observability – necessitating the use of the notion of belief space, the size of which

is exponential in the size of the state space – increases the problem complexity by another exponential, which in the most general case leads to doubly exponential time complexity.

However, interesting phenomena arise from some of the restrictions on the properties the transition descriptions may have. Unlike in the most basic goal reachability problems with deterministic transitions and full observability where the worst-case complexity is in a much bigger extent independent of the properties of transitions, in the partially observable case with deterministic transitions the complexity still decreases further when transitions are also state-independent. This reduction in complexity can be traced to the decrease in the uncertainty about the current state, as each transition can only decrease the amount of uncertainty, not increase it.

The structure of the article is as follows. In Section 2 we briefly discuss the necessary complexity classes and models of computation. The definition of succinct transition systems, the definition of transition systems, different observation models, and the definition of plans and plan objectives are respectively presented in Sections 3.1 3.2, 3.3, 3.4 and 3.5. Reductions between plan objectives, which are the basis of many results in the article, are presented in Section 3.6. Sections 4, 5 and 6 present the complexity results under the three plan objectives, and the results are summarized in Section 7. Connections of these results to discrete event-systems are discussed in Section 8. Finally, related work is discussed in Section 9 and the article is concluded in Section 10.

2. PRELIMINARIES: COMPUTATIONAL COMPLEXITY

In this section we discuss deterministic, nondeterministic and alternating Turing machines (DTMs, NDTMs and ATMs) and define several complexity classes in terms of them. For a detailed introduction to computational complexity see any of the standard textbooks [Balcázar, Díaz, and Gabarró 1988; Balcázar, Díaz, and Gabarró 1990; Papadimitriou 1994].

The definition of ATMs we use is similar to that of Papadimitriou [1994]. **Mitä määritelmää lähimpänä?** Deterministic and nondeterministic Turing machines (DTMs, NDTMs) are a special case of alternating Turing machines.

Definition 1 *An alternating Turing machine is a tuple $\langle \Sigma, Q, \delta, q_0, g \rangle$ where*

- (1) Q is a finite set of states (the internal states of the ATM),
- (2) Σ is a finite alphabet (the contents of tape cells),
- (3) δ is a transition function $\delta : Q \times \Sigma \cup \{ |, \square \} \rightarrow 2^{\Sigma \cup \{ | \}} \times Q \times \{ L, N, R \}$,
- (4) q_0 is the initial state, and
- (5) $g : Q \rightarrow \{ \forall, \exists, \text{accept}, \text{reject} \}$ is a labeling of the states.

The symbols $|$ and \square , the end-of-tape symbol and the blank symbol, in the definition of δ respectively refer to the beginning of the tape and to the end of the tape. It is required that $s = |$ and $m = R$ for all $\langle s, q', m \rangle \in \delta(q, |)$ for any $q \in Q$, that is, at the left end of the tape the movement is always to the right and the end-of-tape symbol $|$ may not be changed. For $s \in \Sigma$ we restrict s' in $\langle s', q', m \rangle \in \delta(q, s)$ to $s' \in \Sigma$, that is, $|$ gets written onto the tape only in the special case when the R/W head is on the end-of-tape symbol. Notice that the transition function is a total function, and the the ATM computation terminated upon reaching an accepting or a rejecting state.

A configuration of an ATM is $\langle q, \sigma, \sigma' \rangle$ where q is the current state, σ is the tape contents left of the R/W head with the rightmost symbol under the R/W head, and σ' is the tape contents strictly right of the R/W head. This is a finite representation of the finite non-blank segment of the tape of the ATM.

We use the notation $\sigma[i]$ for the i th symbol of the string σ . The index of the first (leftmost) element is 0. The number of symbols in string σ is denoted by $|\sigma|$.

The computation of an ATM starts from the initial configuration $\langle q_0, |a, \sigma \rangle$ where $a\sigma$ is the input string of the Turing machine. The symbol ϵ denotes the empty string.

Successor configurations are defined as follows.

- (1) A successor of $\langle q, \sigma a, \sigma' \rangle$ is $\langle q', \sigma, a' \sigma' \rangle$ if $\langle a', q', L \rangle \in \delta(q, a)$.
- (2) A successor of $\langle q, \sigma a, \sigma' \rangle$ is $\langle q', \sigma a', \sigma' \rangle$ if $\langle a', q', N \rangle \in \delta(q, a)$.
- (3) A successor of $\langle q, \sigma a, b \sigma' \rangle$ is $\langle q', \sigma a' b, \sigma' \rangle$ if $\langle a', q', R \rangle \in \delta(q, a)$.
- (4) A successor of $\langle q, \sigma a, \epsilon \rangle$ is $\langle q', \sigma a' \square, \epsilon \rangle$ if $\langle a', q', R \rangle \in \delta(q, a)$.

We write $\langle q, \sigma \rangle \vdash \langle q', \sigma' \rangle$ if the latter is a successor configuration of the former. A configuration $\langle q, \sigma, \sigma' \rangle$ of an ATM is *final* if $g(q) = \text{accept}$ or $g(q) = \text{reject}$.

The acceptance of an input string by an ATM is defined recursively starting from final configurations. A final configuration is 0-accepting if $g(q) = \text{accept}$. Non-final configurations are n -accepting if the state is universal (\forall) and all the successor configurations are m -accepting for some $m < n$ and one of them is $m - 1$ -accepting, or if the state is existential (\exists) and at least one of the successor configurations is $m - 1$ -accepting and for all $n < m - 1$ there are no n -accepting successor configurations. Finally, the ATM accepts a given input string if the initial configuration is n -accepting for some $n \geq 0$. A configuration is *accepting* if it is n -accepting for some $n \geq 0$.

If an ATM accepts a given input string, then we can define an *accepting computation subtree* of the ATM and the input string as a set of accepting configurations that includes the initial configuration, all successor configurations of every accepting \forall -configuration, and for every n -accepting \exists -configuration at least one m -accepting successor configuration such that $m < n$.

A nondeterministic Turing machine is an ATM without universal states. A deterministic Turing machine is an ATM with $|\delta(q, s)| = 1$ for all $q \in Q$ and $s \in \Sigma$.

The complexity classes used in this work are the following. PSPACE is the class of decision problems solvable by deterministic Turing machines that use a number of tape cells bounded by a polynomial on the input length n for all but a finite number of input strings, that is, no configuration in the computation of the Turing machine has more than a polynomial number of non-blank tape cells. For this we use the notation

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

Similarly other complexity classes are defined in terms of the time consumption ($\text{DTIME}(f(n))$) on a deterministic Turing machine, time consumption ($\text{NTIME}(f(n))$) on a nondeterministic Turing machine, or time or space consumption on alternating Turing machines ($\text{ATIME}(f(n))$ or $\text{ASPACE}(f(n))$) [Balcázar, Díaz, and Gabarró 1988; Balcázar, Díaz, and Gabarró 1990].

$$\begin{aligned}
P &= \bigcup_{k \geq 0} \text{DTIME}(n^k) \\
NP &= \bigcup_{k \geq 0} \text{NTIME}(n^k) \\
EXP &= \bigcup_{k \geq 0} \text{DTIME}(2^{n^k}) \\
NEXP &= \bigcup_{k \geq 0} \text{NTIME}(2^{n^k}) \\
EXSPACE &= \bigcup_{k \geq 0} \text{DSpace}(2^{n^k}) \\
2\text{-EXP} &= \bigcup_{k \geq 0} \text{DTIME}(2^{2^{n^k}}) \\
2\text{NEXP} &= \bigcup_{k \geq 0} \text{NTIME}(2^{2^{n^k}}) \\
APSPACE &= \bigcup_{k \geq 0} \text{ASPACE}(n^k) \\
AEXSPACE &= \bigcup_{k \geq 0} \text{ASPACE}(2^{n^k})
\end{aligned}$$

There are many useful connections between complexity classes defined in terms of deterministic and alternating Turing machines [Chandra, Kozen, and Stockmeyer 1981], for example

$$\begin{aligned}
EXP &= \text{APSPACE} \\
2\text{-EXP} &= \text{AEXSPACE}.
\end{aligned}$$

Roughly, an exponential deterministic time bound corresponds to a polynomial alternating space bound.

A problem L is C -hard (where C is any of the complexity classes) if all problems in the class C are polynomial time *many-one reducible* to it; that is, for all problems $L' \in C$ there is a function $f_{L'}$ that can be computed in polynomial time on the size of its input and $f_{L'}(x) \in L$ if and only if $x \in L'$. We say that the function $f_{L'}$ is a translation from L' into L . A problem is C -complete if it belongs to the class C and is C -hard.

3. PRELIMINARIES: PROBLEM DEFINITION

3.1 Transition systems

First we define transition systems in which states are atomic objects and actions are represented as binary relation on the set of states.

Definition 2 A transition system is a 5-tuple $\Pi = \langle S, I, O, G, P \rangle$ where

- (1) S is a finite set of states,
- (2) $I \subseteq S$ is the set of initial states,
- (3) O is a finite set of operators $o \subseteq S \times S$,
- (4) $G \subseteq S$ is the set of goal states, and
- (5) $P = (C_1, \dots, C_n)$ is the partition of S to observational into classes of observationally indistinguishable states satisfying $\bigcup\{C_1, \dots, C_n\} = S$ and $C_i \cap C_j = \emptyset$ for all i, j such that $1 \leq i < j \leq n$.

Making an observation tells which set C_i the current state belongs to. Distinguishing states within a given C_i is not possible by observations.

An operator o can be *applied* in states for which it associates at least one successor state. For these states we use the notation $\text{prec}(o) = \{s \in S \mid sos' \text{ for some } s' \in S\}$. We define

images of states as $img_o(s) = \{s' | sos'\}$ and (weak) preimages of states by $wpreimg_o(s') = \{s | sos'\}$. These can be generalized to sets of states by $img_o(S) = \bigcup_{s \in S} img_o(s)$ and $wpreimg_o(S) = \bigcup_{s \in S} wpreimg_o(s)$. For sequences o_1, \dots, o_n of operators we define $img_{o_1; o_2; \dots; o_n}(S) = img_{o_n}(\dots img_{o_2}(img_{o_1}(S)) \dots)$ and $wpreimg_{o_1; o_2; \dots; o_n}(S) = wpreimg_{o_1}(wpreimg_{o_2}(\dots wpreimg_{o_n}(S)))$. The strong preimage of a set S' of states is the set of states for which all successor states are in S' , defined as $spreimg_o(S') = \{s | s' \in S', sos', img_o(s) \subseteq S'\}$.

Lemma 3 *Images, strong preimages and weak preimages of sets of states are related to each other as follows. Let o be any operator and S and S' any sets of states.*

- (1) $spreimg_o(S) \subseteq wpreimg_o(S)$
- (2) $img_o(spreimg_o(S)) \subseteq S$
- (3) If $S \subseteq S'$ then $img_o(S) \subseteq img_o(S')$.
- (4) $wpreimg_o(S) = spreimg_o(S)$ when o is deterministic.

PROOF. (1)

- (2) Take any $s' \in img_o(spreimg_o(S))$. Hence there is $s \in spreimg_o(S)$ so that sos' . As $s \in spreimg_o(S)$, $img_o(s) \subseteq S$. Since $s' \in img_o(s)$, $s' \in S$.
- (3) Assume $S \subseteq S'$ and $s' \in img_o(S)$. Hence sos' for some $s \in S$ by definition of images. Hence sos' for some $s \in S'$ because $S \subseteq S'$. Hence $s' \in img_o(S')$ by definition of images.
- (4)

□

3.2 Succinct transition systems

It is often more natural to represent states of a transition system as valuations of a set of state variables instead of enumeratively as in Section 3.1. When states are valuations of state variables it is often possible to also represent the transition relations associated with actions more compactly. An action in this setting is a description of how the values of state variables change when the action is taken.

In this section we give a definition of planning in terms of succinct transition systems, similarly to the work by Mundhenk et al. [2000] for planning with POMDPs. Differences to the work by Mundhenk et al. are that we do not use probabilities, and instead of the Boolean circuit presentation of transition matrices of actions we use a slightly simpler representation in which the changes are represented by a simple programs. These programs are constructed from atomic programs, that assign values to state variables, by a composition construct, a conditional construct, and a nondeterministic choice construct. When ignoring nondeterminism, this action description is close to the languages used by the AI planning community, for example the planning language PDDL [Ghallab, Howe, Knoblock, McDermott, Ram, Veloso, Weld, and Wilkins 1998].

The results given in this work do not strongly rely on the exact definition of succinctly represented transition systems. The hardness proofs could be established with a substantially more restricted language, for example without disjunctivity in goals and operator preconditions, and the proofs of membership in certain complexity classes would in many cases be possible with the kind of circuit representation used by Mundhenk et al. [2000].

In our succinct representation states are represented in terms of a set A of Boolean state variables that take the value *true* or *false*. We can form formulae from the state variables and the connectives \vee , \wedge and \neg . The connectives \rightarrow and \leftrightarrow are defined in terms of the other connectives as usual. Each *state* is a valuation of A , that is, a function $s : A \rightarrow \{0, 1\}$. A *literal* is a formula of the form a or $\neg a$ where $a \in A$ is a state variable. The atomic formulae \top and \perp have the truth-values *true* and *false*, respectively.

Definition 4 Let A be a set of state variables. An operator is a pair $\langle c, e \rangle$ where c is a propositional formula over A (the precondition), and e is an effect over A . Effects over A are recursively defined as follows.

- (1) a and $\neg a$ for state variables $a \in A$ are effects over A .
- (2) $e_1 \wedge \dots \wedge e_n$ is an effect over A if e_1, \dots, e_n are effects over A (the special case with $n = 0$ is the empty effect \top .)
- (3) $c \triangleright e$ is an effect over A if c is a formula over A and e is an effect over A .
- (4) $g_1 e_1 | \dots | g_n e_n$ is an effect over A if e_1, \dots, e_n for $n \geq 2$ are effects over A and g_1, \dots, g_n are formulae over A , called the guards.

The guards g_i in the nondeterministic effects determine which effects may become randomly chosen. If the guard g_i is true, then e_i can be the chosen effect. In the rest of the work, if the guard is \top and the effect thus always enabled, the guard may be left out.

Now an *operator* over A is a pair $\langle c, e \rangle$ where c is a formula over A and e is an effect over A . For the *precondition* c of the operator $o = \langle c, e \rangle$ we use the notation $\text{prec}(o)$.

Operators describe a binary relation on the set of states as follows.

Definition 5 (Operator application) Let $\langle c, e \rangle$ be an operator over A . Let s be a state, that is an assignment of truth values to A . The operator is applicable in s if $s \models c$ and the set $[e]_s$ is consistent. The set $[e]_s$ is recursively defined as follows.

- (1) $[a]_s = \{\{a\}\}$ and $[\neg a]_s = \{\{\neg a\}\}$ for $a \in A$.
- (2) $[e_1 \wedge \dots \wedge e_n]_s = \{\bigcup_{i=1}^n f_i \mid f_1 \in [e_1]_s, \dots, f_n \in [e_n]_s\}$.
- (3) $[c' \triangleright e]_s = [e]_s$ if $s \models c'$ and $[c' \triangleright e]_s = \{\emptyset\}$ otherwise.
- (4) $[g_1 e_1 | \dots | g_n e_n]_s = \{c \mid s \models g_1, c \in [e_1]_s\} \cup \dots \cup \{c \mid s \models g_n, c \in [e_n]_s\} \cup \{\emptyset \mid s \models \neg g_1 \wedge \dots \wedge \neg g_n\}$

An operator $\langle c, e \rangle$ induces a binary relation $R\langle c, e \rangle$ on states as follows: two states s and s' are related by $R\langle c, e \rangle$ if $s \models c$ and s' is obtained from s by making the literals in some $f \in [e]_s$ true and retaining the truth-values of state variables not occurring in f .

We define the operations related to images and preimages for operators o in terms of $R(o)$, for example by $w\text{preimg}_o(s) = w\text{preimg}_{R(o)}(s)$.

An operator that does not contain nondeterministic choice (the $|$ operation) in the effect, is *deterministic*. Deterministic operators with conditional effects (the \triangleright operation) are *state-dependent*, because the set of state variables that changes may be dependent on the current state. Other deterministic operators are *state-independent*.

We formally define succinct transition systems.

Definition 6 A succinct transition system is a 5-tuple $\Pi = \langle A, I, O, G, V \rangle$ where

- (1) A is a finite set of state variables,
- (2) I is a formula over A describing the initial states,
- (3) O is a finite set of operators over A ,
- (4) G is a formula over A describing the goal states, and
- (5) $V \subseteq A$ is the set of observable state variables.

Succinct transition systems with $V = A$ are *fully observable*, and succinct transition systems with $V = \emptyset$ are *unobservable*. Without restrictions on V the succinct transition systems are *partially observable*.

We can associate a transition system with every succinct transition system.

Definition 7 Given a succinct transition system $\Pi = \langle A, I, O, G, V \rangle$, let the transition system $F(\Pi) = \langle S, I', O', G', P \rangle$ where

- (1) S is the set of all Boolean valuations of A ,
- (2) $I' = \{s \in S \mid s \models I\}$,
- (3) $O' = \{R(o) \mid o \in O\}$
- (4) $G' = \{s \in S \mid s \models G\}$, and
- (5) $P = (C_1, \dots, C_n)$ where v_1, \dots, v_n for $n = 2^{|V|}$ are all the Boolean valuations of V and $C_i = \{s \in S \mid s(a) = v_i(a) \text{ for all } a \in V\}$ for all $i \in \{1, \dots, n\}$.

The construction of a transition system from a succinct transition system takes polynomial time in the size of the transition system.

Example 8 ■

3.3 Alternative observation models

In our model observability is defined in terms of a subset of the state variables that are observable. More generally, observations could be non-atomic, that is, it could be possible to observe the truth-value of a formula without being able to observe the truth-values of its subformulae. Also, what is observable could be determined by the last operator that has been applied. This would allow formalizing special sensing actions.

In this section we show that the observation model we have adopted is as powerful as models with non-atomic observations and special sensing actions. The reduction from the more general model requires state-dependent effects.

In the more general model, a succinct transition system $\langle A, I, O, G, V \rangle$ does not have a fixed set of observable state variables, but instead V is a mapping from operators O to sets of sets of formulae over A : after applying an operator $o \in O$ the truth-values of the variables $V(o)$ can be observed. Observing the truth-value of a formula ϕ does not mean that the truth-values of its subformulae could be observed.

Let $\Pi = \langle A, I, O, G, V \rangle$ be a succinct transition system. Let β_1, \dots, β_n be the formulas $\beta \in B(o)$ for some $o \in O$. We introduce new auxiliary state variables $a_{\beta_1}, \dots, a_{\beta_n}, z$, and z_i for every $i \in \{1, \dots, m\}$ where m is the number of operators in O . State variable a_β will have the same value as the respective observation β right after an operator that allowed observing it has been applied. The state variables z_i control the application of operators

that evaluate the values of compound observations. Now $C(\Pi) = \langle A, I \wedge z \wedge \neg z_1 \wedge \dots \wedge \neg z_m, O', G, \{a_{\beta_1}, \dots, a_{\beta_n}\} \rangle$ where O' consists of

$$\begin{aligned} &\langle z \wedge c, \neg z \wedge z_i \wedge e \rangle \\ &\langle z_i, z \wedge \neg z_i \wedge \nu \rangle \end{aligned}$$

for every operator $\langle c, e \rangle \in O$ (operator's index is i). The first operator replaces the old operator and the effect ν in the second evaluates the observations β and copies their values to the respective state variables a_β .

$$\nu = \bigwedge \{ (\beta \triangleright a_\beta) \wedge (\neg \beta \triangleright \neg a_\beta) \mid \beta \in B(o) \}$$

The state variable z indicates that any operator can be applied, z_i indicates that operator i has been applied and the corresponding observations are to be evaluated.

Theorem 9 *Let Π be a succinct transition system. Then Π has a plan if and only if $C(\Pi)$ has, and for every plan for Π of length n , there is a plan for $C(\Pi)$ of length $2n$.*

PROOF. We only give a brief proof sketch.

The reduction guarantees that the value of a_β coincides with the value of β after an operator with observation β has been applied.

Translations from plans for Π into plans for $C(\Pi)$ and vice versa do not require changing the structure of the plans, only one operator is interchanged with a sequence of two operators, or vice versa.

Plans for Π can be translated into plans for $C(\Pi)$ by replacing observations of β by observations of a_β and replacing every operator with the corresponding two new operators.

Plans for $C(\Pi)$ can be transformed into plans for Π by replacing observations a_β by β and making the converse replacement of operators. The two operator in $C(\Pi)$ obtained from one operator in Π always appear together, and plans cannot branch between them because the first operator does not have anything observable. \square

3.4 Plans

Plans determine what actions are executed. We formalize plans as a form of directed graphs. Each node is assigned an operator and information on zero or more successor nodes.

Definition 10 *Let $\Pi = \langle A, I, O, G, V \rangle$ be a succinct transition system. A plan for Π is a triple $\langle N, b, l \rangle$ where*

- (1) N is a finite set of nodes,
- (2) $b \subseteq \mathcal{L} \times N$ maps initial states to starting nodes, and
- (3) $l : N \rightarrow O \times 2^{\mathcal{L} \times N}$ is a function that assigns each node n an operator and a set of pairs $\langle \phi, n' \rangle$ where ϕ is a formula over the observable state variables V and $n' \in N$ is a successor node.

Nodes n with $l(n) = \langle o, \emptyset \rangle$ for some $o \in O$ are terminal nodes.

Ignoring the operators and branch formulae in a plan π we can construct a graph $G(\pi) = \langle N, E \rangle$ with $E \subseteq N \times N$ such that $\langle n, n' \rangle \in E$ iff $\langle \phi, n' \rangle \in B$ for $l(n) = \langle o, B \rangle$ and some ϕ . A plan π is acyclic if there is no non-trivial path starting and ending at the same node in $G(\pi)$.

Plan execution starts from a node $n \in N$ and state s such that $\langle \phi, n \rangle \in b$ and $s \models I \wedge \phi$. Execution in node n with $l(n) = \langle o, B \rangle$ proceeds by executing the operator o and then testing for each $\langle \phi, n' \rangle \in l(n)$ whether ϕ is true in all possible current states, and if it is, continuing execution from plan node n' . At most one ϕ may be true for this to be well-defined. Plan execution ends when none of the branch labels matches the current state. In a terminal node plan execution necessarily ends.

We define the satisfaction of plan objectives in terms of the transition system that is obtained when the original transition system is being controlled by a plan, that is, the plan chooses which of the transitions possible in a state is taken. For goal reachability, without unbounded looping it would be required that any maximal path from an initial state has finite length and ends in a goal state. With unbounded looping it would be required that from any state to which there is a path from an initial state that does not visit a goal state there is a path of length ≥ 0 to a goal state.

Definition 11 (Execution graph of a plan) *Let $\langle A, I, O, G, V \rangle$ be a succinct transition system and $\pi = \langle N, b, l \rangle$ be a plan. Define the execution graph of π as a pair $\langle M, E \rangle$ where*

- (1) $M = S \times (N \cup \{\perp\})$, where S is the set of Boolean valuations of A ,
- (2) $E \subseteq M \times M$ has an edge from $\langle s, n \rangle \in S \times N$ to $\langle s', n' \rangle \in S \times N$ if and only if $l(n) = \langle o, B \rangle$ and for some $\langle \phi, n' \rangle \in B$
 - (a) $s' \in \text{img}_o(s)$ and
 - (b) $s' \models \phi$.
- and an edge from $\langle s, n \rangle \in S \times N$ to $\langle s', \perp \rangle$ if and only if
 - (a) $l(n) = \langle o, B \rangle$,
 - (b) $s' \in \text{img}_o(s)$, and
 - (c) there is no $\langle \phi, n' \rangle \in B$ such that $s' \models \phi$.

3.5 Decision problems

There are different types of objectives the plans may have to fulfill. The most basic one, considered in much of AI planning research, is the reachability of a goal state. In this case each plan execution has a finite length. Also problems with infinite plan executions can be considered. A plan does not reach a goal and terminate, but is a continuing process that has to repeatedly reach goal states or avoid visiting bad states. Examples of these are different kinds of maintenance tasks: keep a building clean and transport mail from location A to location B.

We consider two objectives defined in terms of infinite plan executions. One is maintenance goals, the other is repeated reachability. Infinite plan executions are also considered in connection with Markov decision processes and the average-reward and discounted geometric reward objectives [Puterman 1994].

The plan objectives we consider in this work are the following.

- (1) RG reachability (finite horizon)

The objective is to reach one of predefined goal states starting from the initial state.

Length of plan executions in this case is finite but for a given plan the length may still be unbounded if some of the actions are nondeterministic and loops are allowed.

- (2) MG Maintenance goals (infinite horizon)

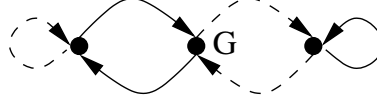


Fig. 1. Reason why plans cannot be defined as mappings from belief states to actions

The objective is to take actions to stay indefinitely within the set of goal states; that is, a plan has to maintain a certain property. Plan executions in this case are infinite.

(3) RRG repeated reachability (infinite horizon)

The objective is to repeatedly reach one of predefined goal states, possibly intermittently visiting non-goal states. After a goal state has been reached, plan execution is continued and a goal state has to be reached again.

In Section 3.6 we show that reachability goals and maintenance goals are special cases of repeated reachability.

Notice that repeated reachability is not a trivial extension of reachability goals, because in the first case plan execution can end only after reaching a belief state consisting of goal states only, whereas in the second case it might never be known whether the current state is a goal state. We illustrate this by an example.

Example 12 Consider the transition graph in Figure 1. The belief state initially and after taking any action consists of all three states. To satisfy the repeated reachability criterion the two actions, one depicted with a dotted line and the other with a continuous line, have to be alternated. In this example Repeated Reachability objective is satisfied although at no point of time is it known that a goal state is occupied. ■

The example shows how the dependencies between consecutive belief states become important in the repeated reachability problem. The belief state, that is the set of possible current states, does not alone carry enough information for deciding whether the objective is fulfilled or not. Also, plans cannot be defined as mappings from belief states (understood as sets of states) to actions, unlike for maintenance or reachability goals.

Definition 13 (Reachability goals RG) A plan $\pi = \langle N, b, l \rangle$ solves a succinct transition system $\langle A, I, O, G, V \rangle$ under the Reachability (RG) criterion if the corresponding execution graph fulfills the following.

For all states s and nodes $n \in N$ such that $\langle \phi, n \rangle \in b$ and $s \models I \wedge G$, for every (s', n') to which there is a path from (s, b) there is a path from (s', n) of length ≥ 0 to some (s'', n') such that $s'' \models G$ and (s'', n') has no successor nodes.

This plan objective with unbounded looping can be interpreted probabilistically. For every nondeterministic choice in an operator we have to assume that each of the alternatives has a non-zero probability. Then for goal reachability, a plan with unbounded looping is simply a plan that has no finite upper bound on the length of its executions, but that with probability 1 eventually reaches a goal state. A non-looping plan also reaches a goal state with probability 1, but there is a finite upper bound on the execution length.

Definition 14 (Maintenance goals MG) A plan $\pi = \langle N, b, l \rangle$ solves a succinct transition system $\langle A, I, O, G, V \rangle$ under the Maintenance (MG) criterion if the corresponding execution graph fulfills the following.

For all states s and s' and plan nodes $n \in N$ and $n' \in N$ such that there is $\langle \phi, n \rangle \in b$ and $s \models I \wedge G$, if there is a path of length ≥ 0 from (s, n) to some (s', n') , then $s' \models G$ and (s', n') has a successor node.

Definition 15 (Repeated reachability goals RRG) A plan $\pi = \langle N, b, l \rangle$ solves a succinct transition system $\langle A, I, O, G, V \rangle$ under the Repeated Reachability (RRG) criterion if the corresponding execution graph fulfills the following.

For all states s and s' and plan nodes $n \in N$ and $n' \in N$ such that there is $\langle \phi, n \rangle \in b$ and $s \models I \wedge G$, if there is a path from (s, n) to (s', n') , then there is a path of length ≥ 1 from (s', n') to (s'', n'') such that $s'' \models G$.

The decision problem addressed in this paper is deciding whether a certain transition system has a plan that satisfies the goal objective.

We consider the three objectives discussed in Section 3.4, reachability goals (RG), repeated reachability goals (RRG) and maintenance goals (MG). Each problem is analyzed under full observability (FO), without observability (UO) and in the general case of partial observability (PO). For transition systems we consider the general definition of nondeterministic operators (ND), their subclass of deterministic operators (D), and the narrowest subclass of deterministic state-independent operators (ID).

Because there are three independent choices made between three alternatives, we have a total of 27 decision problems. When we furthermore distinguish between the two cases of having exactly one initial state and having several, we have 54 decision problems. However, some of the decision problems coincide (for example, for deterministic problems with one initial state observability does not make a difference), and the total number of decision problems to consider is much smaller.

We assign names like PLANSAT-PO-ND-RRG to the decision problems. This problem is the most general problem we consider, with partial observability, nondeterministic operators and the repeated reachability criterion.

We sometimes consider the decision problems of testing existence of acyclic plans. An acyclic plan is a plan like defined in Section 3.4 but that does not have cycles. In these cases the name of the decision problem has the form PLANSAT-PO-ND-acyclicRG. Notice that for unobservable problems with reachability goals we can always restrict to acyclic plans.

3.6 Reductions between decision problems

Reachability goals and maintenance goals are easily reducible to repeated reachability. These reductions are useful for proving lower bounds on plan existence problems for repeated reachability, and for proving upper bounds on plan existence problems for reachability and maintenance goals.

Theorem 16 Let $\Pi = \langle A, I, O, G, V \rangle$ be a succinct transition system. Then Π has a solution for RG if and only if Π' has a solution for RRG, where $\Pi' = \langle A \cup \{a\}, I \wedge \neg a, O \cup \{\langle G, a \rangle, \langle a, \top \rangle\}, a, V \rangle$ and $a \notin A$.

PROOF. Sketch: Assume a plan for Π exists. Any terminal node n can be extended to first apply $\langle G, a \top \rangle$ and then repeat $\langle a, \top \rangle$ indefinitely. This results in a plan that eventually reaches a state satisfying G , then $G \wedge a$, and then stays in goal states indefinitely.

Assume a plan for Π' exists. In this plan we delete all children of nodes that apply $\langle G, a \rangle$ and make those nodes terminal nodes. Hence any execution that reaches a node applying $\langle G, a \rangle$ reaches a goal state. \square

Theorem 17 *Let $\Pi = \langle A, I, O, G, V \rangle$ be a succinct transition system. Then Π has a solution for MG if and only if Π' has a solution for RRG, where $\Pi' = \langle A, I, \{\langle c \wedge G, e \rangle \mid \langle c, e \rangle \in O\}, G, V \rangle$.*

PROOF. Assume there is a plan satisfying the maintenance objective for Π . The plan obtained by replacing every operator $\langle c, e \rangle$ by $\langle c \wedge G, e \rangle$ satisfies the repeated reachability objective for Π' : all operators of the latter plan are always applicable because every state reached during execution satisfies G .

Assume there is a plan satisfying the repeated reachability objective for Π' . Because G is true always when applying an operator, G is true in all states reached during plan execution. Hence the plan obtained by replacing every $\langle c \wedge G, e \rangle$ by $\langle c, e \rangle$ satisfies the maintenance objective. \square

Notice that in both of these proofs all the operators in Π are deterministic (deterministic state-independent) if and only if all those in Π' are deterministic (deterministic state-independent). This allows us to use – under the three classes of operators ND, Dand ID– algorithms and complexity upper bound proofs for repeated reachability also for reachability and maintenance, and complexity lower bound proofs for reachability and maintenance also for repeated reachability.

4. COMPLEXITY OF PLANNING FOR REACHABILITY: LOWER BOUNDS

First we discuss the complexity of plan synthesis for reachability goals. These results yield also lower bounds for the repeated reachability objective, and the hardness results for maintenance goals are in most cases simple modifications of the Turing machine simulations presented in this section.

Theorem 18 ([Bylander 1994]) *Planning with one initial state and deterministic operators, even for state-independent operators (PLANSAT-?-ID-RG), is PSPACE-hard.*

The proof of this theorem is a simulation of polynomial-space deterministic Turing machines. It can be obtained from the proof of Theorem 19 by restricting it to deterministic Turing machines. It is easy to generalize the simulation to nondeterministic NPSPACE=PSPACE Turing machines.

When nondeterministic (with or without probabilities) operators are allowed, the complexity of plan existence problems increases. Littman [1997] reduces the EXP-complete existence problem of winning strategies in the game G_4 [Stockmeyer and Chandra 1979] to the plan existence problem of probabilistic planning. Probabilities do not play a role in G_4 or in the reduction, and hence also the plan existence problem of non-probabilistic planning with fully observability is EXP-hard.

Next we present a proof for the EXP-hardness of conditional planning with full observability by a direct simulation of polynomial-space alternating Turing machines. The result follows from the equality EXP=APSPACE. Proof of Theorem 18 by Bylander shows how PSPACE Turing machines are simulated. For APSPACE we also need to simulate alternation, that is, a configuration of the TM may have several successor configurations, and that there are both \forall and \exists states.

For configurations with \forall states all successor configurations must be accepting (terminal or non-terminal) configurations. For configurations with \exists states at least one successor configuration must be an accepting (terminal or non-terminal) configuration. Both of these requirements can be represented in the nondeterministic planning problem.

The transitions from a configuration with a \forall state will correspond to one nondeterministic operator. That all successor configurations must be accepting (terminal or non-terminal) configurations corresponds to requirement in planning that from all successor states of a state a goal state must be reached.

Every transition from a configuration with \exists state will correspond to a deterministic operator, that is, the transition may be chosen, as only one of the successor configurations needs to be accepting.

Theorem 19 *The problem of testing the existence of an acyclic plan for problem instances with full observability (PLANSAT-FO-ND-acyclicRG) is EXP-hard, even with the restriction to only one initial state.*

PROOF. Let $\langle \Sigma, Q, \delta, q_0, g \rangle$ be any alternating Turing machine with a polynomial space bound $p(x)$. Let σ be an input string of length n . We construct a succinct transition system $\langle A, I, O, G, A \rangle$ with full observability for simulating the Turing machine. The succinct transition system has a size that is polynomial in the size of the description of the Turing machine and the input string.

The set A of state variables in the succinct transition system consists of

- (1) $q \in Q$ for denoting the states of the TM,
- (2) s_i for every symbol $s \in \Sigma \cup \{ |, \square \}$ and tape cell $i \in \{0, \dots, p(n)\}$, and
- (3) h_i for the positions of the R/W head $i \in \{0, \dots, p(n) + 1\}$.

The initial state of the succinct transition system represents the initial configuration of the TM. The corresponding formula I is the conjunction of the following literals.

- (1) q_0
- (2) $\neg q$ for all $q \in Q \setminus \{q_0\}$.
- (3) s_i for all $s \in \Sigma$ and $i \in \{1, \dots, n\}$ such that i th input symbol is s .
- (4) $\neg s_i$ for all $s \in \Sigma$ and $i \in \{1, \dots, n\}$ such that i th input symbol is not s .
- (5) $\neg s_i$ for all $s \in \Sigma$ and $i \in \{0, n + 1, n + 2, \dots, p(n)\}$.
- (6) \square_i for all $i \in \{n + 1, \dots, p(n)\}$.
- (7) $\neg \square_i$ for all $i \in \{0, \dots, n\}$.
- (8) $|_0$
- (9) $\neg |_i$ for all $i \in \{1, \dots, p(n)\}$
- (10) h_1
- (11) $\neg h_i$ for all $i \in \{0, 2, 3, 4, \dots, p(n) + 1\}$

The goal is the following formula.

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

Next we define the set O of operators corresponding to transitions of the ATM. We have to distinguish between transitions for universal \forall and existential \exists states.¹ For a given input

¹There are no transitions for accepting and rejecting states.

symbol and a \forall state, the transitions from that state correspond to one nondeterministic operator, whereas for a given input symbol and an \exists state each transition corresponds to a different deterministic operator.

We first define the effects of the operators. For all $\langle s, q \rangle \in (\Sigma \cup \{[, \square\}) \times Q$, $i \in \{0, \dots, p(n)\}$ and $\langle s', q', m \rangle \in (\Sigma \cup \{[, \square\}) \times Q \times \{L, N, R\}$ define $\tau_{s,q,i}(s', q', m) = \alpha \wedge \kappa \wedge \theta$ where the effects α , κ and θ are defined below.

The effect α describes the changes to the tape symbol at the R/W head. If $s = s'$ then $\alpha = \top$ as nothing on the tape changes. Otherwise, $\alpha = \neg s_i \wedge s'_i$ to denote that the new symbol in the i th tape cell is s' and not s .

The effect κ describes the change to the state of the TM. We define $\kappa = \neg q$ when $i = p(n)$ and $m = R$ so that when the space bound is violated, no accepting state is reached. Otherwise, that is, if $i < p(n)$ or $m \neq R$, define κ as follows. If $q = q'$ then $\kappa = \top$ and otherwise $\kappa = \neg q \wedge q'$.

The effect θ describes the movement of the R/W head. Either there is movement to the left, no movement, or movement to the right.

$$\theta = \begin{cases} \neg h_i \wedge h_{i-1} & \text{if } m = L \\ \top & \text{if } m = N \\ \neg h_i \wedge h_{i+1} & \text{if } m = R \end{cases}$$

By definition of TMs, movement at the left end of the tape is always to the right. Also, we have the state variable $h_{p(n)+1}$ for R/W head position $p(n) + 1$ and moving to that configuration is possible (violating the space bound), but there are no operators for transitions from that configuration.

Now we define the operators based on the above effects. Operators for existential states q with $g(q) = \exists$ and for universal states q with $g(q) = \forall$ differ. Let $\langle s, q \rangle \in (\Sigma \cup \{[, \square\}) \times Q$, $i \in \{0, \dots, p(n)\}$ and $\delta(s, q) = \{\langle s_1, q_1, m_1 \rangle, \dots, \langle s_k, q_k, m_k \rangle\}$.

If $g(q) = \exists$, then define k deterministic operators

$$\begin{aligned} o_{s,q,i,1} &= \langle h_i \wedge s_i \wedge q, \tau_{s,q,i}(s_1, q_1, m_1) \rangle \\ o_{s,q,i,2} &= \langle h_i \wedge s_i \wedge q, \tau_{s,q,i}(s_2, q_2, m_2) \rangle \\ &\vdots \\ o_{s,q,i,k} &= \langle h_i \wedge s_i \wedge q, \tau_{s,q,i}(s_k, q_k, m_k) \rangle. \end{aligned}$$

For a given current symbol s , state q and tape cell i a plan can apply one of these operators, that is, the plan determines which \exists transition is chosen.

If $g(q) = \forall$, then define one nondeterministic operator

$$\begin{aligned} o_{s,q,i} &= \langle h_i \wedge s_i \wedge q, (\tau_{s,q,i}(s_1, q_1, m_1) | \\ &\quad \tau_{s,q,i}(s_2, q_2, m_2) | \\ &\quad \vdots \\ &\quad \tau_{s,q,i}(s_k, q_k, m_k)) \rangle. \end{aligned}$$

That is, the \forall transition is chosen nondeterministically.

For establishing a connection between the ATM and the planning we use the following definitions. Let $c = \langle q, \sigma, \sigma' \rangle$ be a configuration of the ATM. Define the state $z = CS(c)$ as follows.

$$(1) \quad z(q) = 1$$

- (2) $z(q') = 0$ for all $q' \in Q \setminus \{q\}$
- (3) $z(s_i) = 1$ iff $(\sigma\sigma')[i] = s$, for all $s \in \Sigma$ and $i \in \{0, \dots, |\sigma\sigma'| - 1\}$
- (4) $z(s_i) = 0$ for all $s \in \Sigma$ and $i \in \{|\sigma\sigma'|, \dots, p(n)\}$
- (5) $z(|_0) = 1$
- (6) $z(|_i) = 0$ for all $i \in \{1, \dots, p(n)\}$
- (7) $z(\sqcap_i) = 0$ for all $i \in \{0, \dots, |\sigma\sigma'| - 1\}$
- (8) $z(\sqcap_i) = 1$ for all $i \in \{|\sigma\sigma'|, \dots, p(n)\}$
- (9) $z(h_i) = 1$ iff $i = |\sigma|$, for all $i \in \{0, \dots, p(n) + 1\}$

Claim A: For configurations $\langle q, \sigma_1, \sigma_2 \rangle$ and $\langle q', \sigma'_1, \sigma'_2 \rangle$ such that $q, q' \in Q$ and $g(q) = \exists, \langle q, \sigma_1, \sigma_2 \rangle \vdash \langle q', \sigma'_1, \sigma'_2 \rangle$ if and only if $\{CS(\langle q', \sigma'_1, \sigma'_2 \rangle)\} = \text{img}_o(CS(\langle q, \sigma_1, \sigma_2 \rangle))$ for some $o \in O$.

Claim B: For configurations $\langle q, \sigma_1, \sigma_2 \rangle$ such that $g(q) = \forall, \{CS(\langle q', \sigma'_1, \sigma'_2 \rangle) \mid \langle q, \sigma_1, \sigma_2 \rangle \vdash \langle q', \sigma'_1, \sigma'_2 \rangle\} = \text{img}_o(CS(\langle q, \sigma_1, \sigma_2 \rangle))$ for some $o \in O$ and operators in $O \setminus \{o\}$ are not applicable in $CS(\langle q, \sigma_1, \sigma_2 \rangle)$.

We prove that succinct transition system $\langle A, I, O, G, A \rangle$ has a plan if and only if the alternating Turing machine $\langle \Sigma, Q, \delta, q_0, g \rangle$ with input string σ accepts without violating the space bound.

If the Turing machine violates the space bound, the state variable $h_{p(n)+1}$ becomes true and goal states cannot be reached because no further operator will be applicable.

Otherwise, we can inductively extract from a computation tree of an accepting ATM a conditional plan that always reaches a goal state, and vice versa.

First we show how accepting computation trees are mapped to plans with execution graphs satisfying the RG criterion so that all paths in the graph are finite. We take the configurations of the ATM to be the *nodes* in the plan. The construction proceeds inductively.

Base case $d = 0$: 0-accepting (final) configurations $c = \langle q, \sigma_1, \sigma_2 \rangle$ are mapped to plan nodes c with $l(c) = \langle \text{NOOP}, \emptyset \rangle$. In the execution graph there are two corresponding nodes: $\langle CS(c), c \rangle$, where $CS(c)$ is a goal state, followed by $\langle CS(c), \perp \rangle$. Hence from the node $\langle CS(c), c \rangle$ there is a path of length 1 to a terminal node.

Inductive case $d \geq 1$:

- (1) Let c be a d -accepting \exists -configuration with current symbol s , current state q and R/W head position i , and a $d - 1$ -accepting successor configuration c' reached by transition $\langle s', q', m \rangle$.

The configuration c is mapped to a plan node c with $l(c) = \langle o_{s,q,i,j}, \{\langle \top, c' \rangle\} \rangle$ where s is the current symbol, q is the current state, and j is the index of the transition $\langle s', q', m \rangle$.

By Claim A, in the execution graph there is an edge from $\langle CS(c), c \rangle$ to $\langle CS(c'), c' \rangle$.

By induction assumption all paths in the execution graph from $\langle CS(c'), c' \rangle$ have a finite length and end in a node corresponding to a goal state. Hence all paths starting from $\langle CS(c), c \rangle$ have length at most $d + 1$ and end in a goal node.

- (2) Let c be a d -accepting \forall -configuration with successor configurations c_1, \dots, c_n , R/W head position i , current symbol s , and state q . Let $\langle s^1, q_1, m_1 \rangle, \dots, \langle s^n, q_n, m_n \rangle$ be the transitions from c that respectively reach c_1, \dots, c_n . Now for the plan node c we define $l(c) = \langle o_{s,q,i}, \{\langle s_i^1 \wedge q_1 \wedge l_1, c_1 \rangle, \dots, \langle s_i^n \wedge q_n \wedge l_n, c_n \rangle\} \rangle$, where for every $j \in \{1, \dots, n\}$, $l_j = h_{i+1}$ if $m_j = R$, $l_j = h_{i-1}$ if $m_j = L$ and otherwise $l_j = h_i$.

By Claim B $img_{o_{s,q,i}}(CS(c)) = \{CS(c_1), \dots, CS(c_n)\}$. These states differ with respect to state variables representing the symbol that was written, the state of the TM, and the location of the R/W head. Branching in $l(c)$ distinguishes between all these states. Hence in the execution graph the edges from $\langle CS(c), c \rangle$ are to nodes $\langle CS(c_1), c_1 \rangle, \dots, \langle CS(c_n), c_n \rangle$.

By induction assumption all paths from the latter nodes have length $\leq d$ and end in a node corresponding to a goal state, with at least one of the paths of length d . Hence all paths starting from $\langle CS(c), c \rangle$ have length $d + 1$ and end in a goal node.

Finally, the plan is $\langle N, b, l \rangle$ where N is the set of accepting configurations, $b = \{\langle \top, \langle q_0, |a, \sigma' \rangle \rangle\}$ where the input string $\sigma = a\sigma'$. All paths in the execution graph from the unique starting node $\langle CS(\langle q_0, |a, \sigma' \rangle), \langle q_0, |a, \sigma' \rangle \rangle$ are finite and end in a node corresponding to a goal state.

Then we show how plans for the succinct transition system are mapped to accepting computation trees of the ATM. Notice that we only consider acyclic plans. The proof is extended to cyclic plans in Section 4.3. The acyclicity of the plans entails the acyclicity of the execution graphs.

Given an execution graph satisfying the reachability criterion, we construct an accepting computation tree for the Turing machine. First we give a mapping from states to configurations. This mapping is defined only for states v that correspond to configurations in the sense that $v(q) = 1$ for exactly one $q \in Q$ and of the state variables representing the tape contents and the location of the R/W head exactly one is true for any tape location. Any state reachable from the unique initial state of the planning problem satisfies these requirements. Let $SC(v) = \langle q, \sigma, \sigma' \rangle$ where

- (1) q is the state in Q such that $v(q) = 1$,
- (2) σ is the sequence s^1, \dots, s^n of symbols such that $v(s_0^i) = 1$ for all $i \in \{0, \dots, n-1\}$ and $v(h_n) = 1$,
- (3) σ' the empty sequence ϵ if $v(\square_n) = 1$ or $v(\square_{n+1}) = 1$ and otherwise the sequence s'^1, \dots, s'^m of symbols such that $v(s_{i+n}^i) = 1$ for all $i \in \{1, \dots, m\}$. \dots , $v(s_{n+m-1}^n) = 1$ and $v(h_n) = 1$ $v(\square_{n+m-1}) = 0$ and $v(\square_{n+m}) = 1$.

Notice for any ATM configuration c we have $SC(CS(c)) = c$.

Let s_I be the unique state such that $s_I \models I$. Let $\langle N, b, l \rangle$ be a plan for the succinct transition system. Let R be the set of nodes of the execution graph to which there is a path from the initial node $\langle s_I, n \rangle$.

We claim that $T = \{SC(s) | \langle s, n \rangle \in M\}$ is an accepting computation tree of the Turing machine.

The proof is by induction on the length of the shortest path in the execution graph from a node to a terminal node.

Base case $d = 0$: Assume that for $c = \langle q, \sigma, \sigma' \rangle = SC(s) \in T$ the node $\langle s, n \rangle$ is a terminal node in the execution graph. Because the node has no successors it must be that s is a goal state. Hence q is an accepting state and c is an accepting terminal configuration.

Inductive case $d \geq 1$: Assume that for $c = \langle q, \sigma, \sigma' \rangle = SC(s) \in T$ the node $\langle s, n \rangle$ is a node in the execution graph for which the shortest path to a terminal node has length d .

If c is an \exists -configuration, then by Claim A **sanooko väite A täsmälleen tämän?** $\langle s, n \rangle$ has exactly one successor node $\langle s', n' \rangle$ with a distance $d - 1$ to a terminal node. By the

induction hypothesis $SC(\langle s', n' \rangle)$ is an accepting configuration of the Turing machine. Hence also c is an accepting configuration.

If c is a \forall -configuration, then by Claim B **sanooko väite B täsmälleen tämän?** $\langle s, n \rangle$ has successor nodes $\langle s_1, n_1 \rangle, \dots, \langle s_k, n_k \rangle$ with all having distance $\leq d - 1$ to a terminal node. **tässä suunnitelman oletetaan haarautuvan, vaikka se ei välttämättä sitä tee!!** By the induction hypothesis $SC(\langle s_1, n_1 \rangle), \dots, SC(\langle s_k, n_k \rangle)$ are accepting configurations of the Turing machine. Hence also c is an accepting configuration.

So, because all alternating Turing machines with a polynomial space bound can be in polynomial time translated into a nondeterministic planning problem, all decision problems in APSPACE are polynomial time many-one reducible to nondeterministic planning, and the plan existence problem is APSPACE-hard and consequently EXP-hard. \square

4.1 Planning without observability

The plan existence problem of conditional planning with unobservability is more complex than that of conditional planning with full observability. We give a new EXPSPACE-hardness proof for a result first proved by Haslum and Jonsson [2000] and later extend this proof to obtain 2-EXP-hardness of the more general partially observable case.

We show the EXPSPACE-hardness by a direct simulation of exponential space Turing machines. The first problem is the representation of the exponentially long tape. In the PSPACE- and APSPACE-hardness proofs of deterministic planning and conditional planning with full observability, it is possible to represent the polynomial number of tape cells as state variables of the planning problem. With an exponential space bound an exponential number of state variables would be needed, and such a reduction would not be possible in polynomial time.

Hence we have to find a more clever way of encoding the tape contents. It turns out that we can use the uncertainty about the initial state for this purpose. When an execution of the plan that simulates the Turing machine is started, we randomly choose one of the tape cells to be the *watched* tape cell. This is the only cell of the tape for which the contents are represented in the state variables. Of all changes to the working tape the simulated TM makes, only the changes of the watched tape cell are reflected in the state variables.

For guaranteeing that the plan corresponds to a simulation of the Turing machine it is tested whether the transition the plan makes when the current tape cell is the watched tape cell is the one that assumes the current symbol to be the one that is stored in the state variables. If it is not, the plan is not a valid plan. Because the watched tape cell could be any of the exponential number of tape cells, all the transitions the plan makes are guaranteed to correspond to the contents of the current tape cell of the Turing machine, so if the plan does not simulate the Turing machine, the plan is not guaranteed to reach the goal states.

The proof uses several initial states and unobservability. Several initial states are needed for selecting the watched tape cell, and unobservability is needed so that the plan cannot cheat: if the plan can determine what the current tape cell is, it could choose transitions correspond to the Turing machine transitions only on the watched tape cell. Alternatively, we can have only one initial state and select the watched tape cell by one nondeterministic operator in the beginning.

Theorem 20 *The problems of testing the existence of a plan for problem instances with unobservability and deterministic operators (PLANSAT-UO-D-RG) and with nondeterministic operators and one initial state (PLANSAT-UO-ND-RG-1) are EXPSPACE-hard.*

PROOF. We give a simulation of deterministic Turing machines with an exponential space bound. Nondeterministic Turing machines could be simulated for a NEXPSPACE-hardness proof, but this additional generality is not interesting because $\text{EXPSPACE} = \text{NEXPSPACE}$.

Below we give the proof for multiple initial states and deterministic operators, but we could instead use one initial state and force the plan to first apply a special operator that nondeterministically generates the required successor states, so the theorem holds under either assumption.

Let $\langle \Sigma, Q, \delta, q_0, g \rangle$ be any deterministic Turing machine with an exponential space bound $e(n)$. Let σ be an input string of length n . We denote the i th symbol of σ by σ_i .

The Turing machine may use space $e(n)$, and for encoding numbers from 0 to $e(n) + 1$ corresponding to the tape cells we need $m = \lceil \log_2(e(n) + 2) \rceil$ Boolean state variables.

We construct a succinct transition system without observability for simulating the Turing machine. The succinct transition system has a size that is polynomial in the size of the description of the Turing machine and the input string.

Unlike in the proof of Theorem 19 we cannot represent the contents of the tape by having a state variable for every tape cell because an exponential number of state variables would be needed, and the reduction would take exponential time.

It turns out that it suffices to keep track of only one tape cell (which we will call the *watched tape cell*) that is randomly chosen in the beginning of every execution of the plan.

The set A of state variables in the succinct transition system consists of

- (1) q for denoting the states of the TM,
- (2) w_i for $i \in \{0, \dots, m-1\}$ for the watched tape cell $i \in \{0, \dots, e(n)\}$,
- (3) s for every symbol $s \in \Sigma \cup \{|\, \square\}$ for the contents of the watched tape cell,
- (4) h_i for $i \in \{0, \dots, m-1\}$ for the position of the R/W head $i \in \{0, \dots, e(n) + 1\}$.

The uncertainty in the initial state is about which tape cell is the watched one. Otherwise the formula encodes the initial configuration of the TM, and it is the conjunction of the following formulae.

- (1) q_0
- (2) $\neg q$ for all $q \in Q \setminus \{q_0\}$.
- (3) Initialization of the state variables for the contents of the watched tape cell.

$$\begin{aligned} | &\leftrightarrow (w = 0) \\ \square &\leftrightarrow (w > n) \\ s &\leftrightarrow \bigvee_{i \in \{1, \dots, n\}, \sigma_i = s} (w = i) \text{ for all } s \in \Sigma \end{aligned}$$

- (4) $h = 1$ for the initial position of the R/W head.

So the initial state formula allows any values for state variables w_i and the values of the state variables $s \in \Sigma$ are determined on the basis of the values of w_i and the input string. The expressions $w = i$, $w > i$ denote the obvious formulae for testing integer equality and inequality of the numbers encoded by w_0, \dots, w_{m-1} . Later we will also use effects

$h := h + 1$ and $h := h - 1$ that represent incrementing and decrementing the number encoded by h_0, \dots, h_{m-1} .

The goal is the following formula.

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

To define the operators, we first define effects corresponding to transitions. These are similar to those in proof of Theorem 19 with some key differences related to the exponentially long work tape.

For all $\langle s, q \rangle \in (\Sigma \cup \{|\}, \square\}) \times Q$ and $\langle s', q', m \rangle \in (\Sigma \cup \{|\}) \times Q \times \{L, N, R\}$ define the effect $\tau_{s,q}(s', q', m)$ as $\alpha \wedge \kappa \wedge \theta$ where α , κ and θ are defined below.

The effect α describes what happens to the tape symbol under the R/W head. If $s = s'$ then $\alpha = \top$ as nothing on the tape changes. Otherwise, $\alpha = ((h = w) \triangleright (\neg s \wedge s'))$ to denote that the new symbol in the watched tape cell is s' and not s .

The effect κ describes the change to the state of the TM. If the R/W head movement is to the right we define $\kappa = \neg q \wedge ((h < e(n)) \triangleright q')$ if $q \neq q'$ and $(h = e(n)) \triangleright \neg q$ otherwise. This prevents reaching an accepting state if the space bound is violated: no further operator applications are possible. If the movement is left or there is no movement we define $\kappa = \neg q \wedge q'$ if $q \neq q'$ and \top otherwise.

The effect θ describes the movement of the R/W head. Either there is movement to the left, no movement, or movement to the right.

$$\theta = \begin{cases} h := h - 1 & \text{if } m = L \\ \top & \text{if } m = N \\ h := h + 1 & \text{if } m = R \end{cases}$$

By definition of TMs, movement at the left end of the tape is always to the right.

Now, these effects $\tau_{s,q}(s', q', m)$ which represent possible transitions are used in the operators that simulate the TM. Let $\langle s, q \rangle \in (\Sigma \cup \{|\}, \square\}) \times Q$ and $\delta(s, q) = \{\langle s', q', m \rangle\}$.

There are no universal states and computation from accepting and rejecting states does not proceed further. Hence we only define for states $q \in Q$ such that $g(q) = \exists$ the operator

$$o_{s,q} = \langle ((h \neq w) \vee s) \wedge q, \tau_{s,q}(s', q', m) \rangle.$$

Given an accepting computation tree of the Turing machine, we will show that there is a corresponding execution graph that satisfies the RG criterion.

For establishing a connection between the TM and the planning problem consider the following definitions. Let $c = \langle q, \sigma, \sigma' \rangle$ be a configuration of the TM. Given $i \in \{0, \dots, e(n)\}$ that indicates the watched tape cell, define the state $z = CS_i(c)$ as follows.

- (1) $z(q) = 1$
- (2) $z(q') = 0$ for all $q' \in Q \setminus \{q\}$
- (3) $z(s) = 1$ iff $i \in \{0, \dots, |\sigma\sigma'| - 1\}$ and $(\sigma\sigma')[i] = s$, for all $s \in \Sigma$
- (4) $z(|) = 1$ iff $i = 0$
- (5) $z(\square) = 0$ iff $i \in \{0, \dots, |\sigma\sigma'| - 1\}$
- (6) $z(w_j) = 1$ iff j th bit of i is 1, for all $j \in \{0, \dots, m - 1\}$
- (7) $z(h_j) = 1$ iff j th bit of $|\sigma|$ is 1, for all $j \in \{0, \dots, m - 1\}$

We construct a plan and show inductively that in its execution graph all maximal paths from a node corresponding to a d -accepting configuration lead to a terminal node corresponding to a goal state and have length $d + 1$.

Base case $d = 0$: Consider a 0-accepting configuration c . We map c to a plan node c with $l(c) = \langle \text{NOOP}, \emptyset \rangle$. In the execution graph there are several terminal nodes corresponding to this plan node, two for every possible watched tape cell: $\langle CS_i(c), c \rangle$ for all $i \in \{0, \dots, e(n)\}$ followed by $\langle CS_i(c), \perp \rangle$. The state $CS_i(c)$ is a goal state for every $i \in \{1, \dots, e(n)\}$. Hence the path from $\langle CS_i(c), c \rangle$ to a goal node has length 1.

Inductive case $d \geq 1$: Consider a d -accepting configuration c . It has a $d - 1$ -accepting configuration c' . By induction hypothesis for every $i \in \{0, \dots, e(n)\}$ the maximal path in the execution graph from $\langle CS_i(c'), c' \rangle$ have length d and leads to a terminal node $\langle CS_i(c''), \perp \rangle$ such that $CS_i(c'')$ is a goal state.

We can verify **perustuen operaattoreiden määritelmään** that there is an operator o such that $\text{img}_o(CS_i(c)) = \{CS_i(c')\}$. Hence in the execution graph there is a path with the above properties also starting from every node $\langle CS_i(c), c \rangle$ with $i \in \{0, \dots, e(n)\}$.

Hence the RG criterion is satisfied and the operator sequence extracted from the computation tree is a plan.

Now assume a plan o_1, \dots, o_n exists. We construct a sequence c_0, \dots, c_n of configurations that form the accepting computation tree of the Turing machine.

Let $c_0 = \langle q_0, |a, \sigma' \rangle$ where the input string $\sigma = a\sigma'$ is the input string.

$c_i = NC_{o_i}(c_{i-1})$ where

$$NC_{o_{s,q}}(\langle q, \sigma a, \sigma' \rangle) = \begin{cases} \langle q', \sigma, a' \sigma' \rangle & \text{if } \delta(s, q) = \langle a', q', L \rangle \\ \langle q', \sigma a', \sigma' \rangle & \text{if } \delta(s, q) = \langle a', q', N \rangle \\ \langle q', \sigma a' b, \sigma'' \rangle & \text{if } \delta(s, q) = \langle a', q', R \rangle \text{ and } \sigma' = b\sigma'' \\ \langle q', \sigma a' \square, \epsilon \rangle & \text{if } \delta(s, q) = \langle a', q', R \rangle \text{ and } \sigma' = \epsilon \end{cases}$$

kesken

It is easy to verify that the planning problem simulates the TM assuming that when operator $o_{s,q}$ is executed the current tape symbol is indeed s . So assume that some $o_{s,q}$ is the first operator that misrepresents the tape contents and that $h = c$ for some tape cell location c . Now there is an execution of the plan so that $w = c$. On this execution the precondition of $o_{s,q}$ is not satisfied, and the plan is not executable. Hence a valid plan cannot contain operators that misrepresent the tape contents. \square

4.2 Planning with partial observability

We show that the plan existence problem of the general conditional planning problem with partial observability is 2-EXP-hard. The hardness proof is by a simulation of AEXPSPACE=2-EXP Turing machines.

The hardness proof is an extension of both the EXP-hardness proof of Theorem 19 and of the EXPSPACE-hardness proof of Theorem 20. From the first proof we have the simulation of alternation, and from the second proof the simulation of an exponentially long tape.

Theorem 21 *The problem of testing the existence of an acyclic plan for problem instances with partial observability (PLANSAT-PO-ND-acyclicRG) is 2-EXP-hard. This holds even when there is only one initial state.*

PROOF. Let $\langle \Sigma, Q, \delta, q_0, g \rangle$ be any alternating Turing machine with an exponential space bound $e(x)$. Let σ be an input string of length n . We denote the i th symbol of σ by σ_i .

The Turing machine may use space $e(n)$, and for encoding numbers from 0 to $e(n) + 1$ corresponding to the tape cells we need $m = \lceil \log_2(e(n) + 2) \rceil$ Boolean state variables.

We construct a succinct transition system with full observability for simulating the Turing machine. The succinct transition system has a size that is polynomial in the size of the description of the Turing machine and the input string.

We cannot have a state variable for every tape cell because the reduction from Turing machines to planning would not be polynomial time. It turns out that it suffices to keep track of only one tape cell (which we will call the *watched tape cell*) that is randomly chosen in the beginning of every execution of the plan.

The set A of state variables in the succinct transition system consists of

- (1) $q \in Q$ for denoting the states of the TM,
- (2) w_i for $i \in \{0, \dots, m - 1\}$ for the watched tape cell $i \in \{0, \dots, e(n)\}$,
- (3) s for every symbol $s \in \Sigma \cup \{|\, \square\}$ for the contents of the watched tape cell,
- (4) s^* for every $s \in \Sigma \cup \{|\}$ for the symbol last written (important for nondeterministic transitions),
- (5) L, R and N for the last movement of the R/W head (important for nondeterministic transitions), and
- (6) h_i for $i \in \{0, \dots, m - 1\}$ for the position of the R/W head $i \in \{0, \dots, e(n) + 1\}$.

The observable state variables are L, N and $R, q \in Q$, and s^* for $s \in \Sigma$. These are needed by the plan to decide how to proceed execution after a nondeterministic transition with a \forall state.

The uncertainty in the initial state is about which tape cell is the watched one. Otherwise the formula encodes the initial configuration of the TM, and it is the conjunction of the following formulae.

- (1) q_0
- (2) $\neg q$ for all $q \in Q \setminus \{q_0\}$.
- (3) $\neg s^*$ for all $s \in \Sigma \cup \{|\}$.
- (4) Formulae for having the contents of the watched tape cell in state variables $\Sigma \cup \{|\, \square\}$.

$$\begin{aligned} | &\leftrightarrow (w = 0) \\ \square &\leftrightarrow (w > n) \\ s &\leftrightarrow \bigvee_{i \in \{1, \dots, n\}, \sigma_i = s} (w = i) \text{ for all } s \in \Sigma \end{aligned}$$

- (5) $h = 1$ for the initial position of the R/W head.

So the initial state formula allows any values for state variables w_i and the values of the state variables $s \in \Sigma$ are determined on the basis of the values of w_i . The expressions $w = i, w > i$ denote the obvious formulae for testing integer equality and inequality of the numbers encoded by w_0, w_1, \dots . Later we will also use effects $h := h + 1$ and $h := h - 1$ that represent incrementing and decrementing the number encoded by h_0, h_1, \dots .

The goal is the following formula.

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

Next we define the operators. All the transitions may be nondeterministic, and the important thing is whether the transition is for a \forall state or an \exists state. For a given input symbol and a \forall state, the transition corresponds to one nondeterministic operator, whereas for a given input symbol and an \exists state the transitions corresponds to a set of deterministic operators.

To define the operators, we first define effects corresponding to all possible transitions.

For all $\langle s, q \rangle \in (\Sigma \cup \{|\}, \square\}) \times Q$ and $\langle s', q', m \rangle \in (\Sigma \cup \{|\}) \times Q \times \{L, N, R\}$ define the effect $\tau_{s,q}(s', q', m)$ as $\alpha \wedge \kappa \wedge \theta$ where the effects α , κ and θ are defined as follows.

The effect α describes how the current tape symbol changes. If $s = s'$ then $\alpha = \top$ as nothing on the tape changes. Otherwise, $\alpha = ((h = w) \triangleright (\neg s \wedge s')) \wedge s'^* \wedge \neg s^*$ to denote that the new symbol in the watched tape cell is s' and not s , and to make it possible for the plan to detect which symbol was written to the tape by the possibly nondeterministic transition.

The effect κ describes the change to the state of the TM. Again, either the state changes or does not, so $\kappa = \neg q \wedge q'$ if $q \neq q'$ and \top otherwise. If R/W head movement is to the right we define $\kappa = \neg q \wedge ((h < e(n)) \triangleright q')$ if $q \neq q'$ and $(h = e(n)) \triangleright \neg q$ otherwise. This prevents reaching an accepting state if the space bound is violated: no further operator applications are possible.

The effect θ describes the movement of the R/W head. Either there is movement to the left, no movement, or movement to the right. Hence

$$\theta = \begin{cases} (h := h - 1) \wedge L \wedge \neg N \wedge \neg R & \text{if } m = L \\ N \wedge \neg L \wedge \neg R & \text{if } m = N \\ (h := h + 1) \wedge R \wedge \neg L \wedge \neg N & \text{if } m = R \end{cases}$$

By definition of TMs, movement at the left end of the tape is always to the right.

Now, these effects $\tau_{s,q}(s', q', m)$ which represent possible transitions are used in the operators that simulate the ATM. Operators for existential states $q, g(q) = \exists$ and for universal states $q, g(q) = \forall$ differ. Let $\langle s, q \rangle \in (\Sigma \cup \{|\}, \square\}) \times Q$ and $\delta(s, q) = \{\langle s_1, q_1, m_1 \rangle, \dots, \langle s_k, q_k, m_k \rangle\}$.

If $g(q) = \exists$, then define k deterministic operators

$$\begin{aligned} o_{s,q,1} &= \langle ((h \neq w) \vee s) \wedge q, \tau_{s,q}(s_1, q_1, m_1) \rangle \\ o_{s,q,2} &= \langle ((h \neq w) \vee s) \wedge q, \tau_{s,q}(s_2, q_2, m_2) \rangle \\ &\vdots \\ o_{s,q,k} &= \langle ((h \neq w) \vee s) \wedge q, \tau_{s,q}(s_k, q_k, m_k) \rangle \end{aligned}$$

That is, the plan determines which transition is chosen.

If $g(q) = \forall$, then define one nondeterministic operator

$$\begin{aligned} o_{s,q} &= \langle ((h \neq w) \vee s) \wedge q, (\tau_{s,q}(s_1, q_1, m_1) | \\ &\quad \tau_{s,q}(s_2, q_2, m_2) | \\ &\quad \vdots \\ &\quad \tau_{s,q}(s_k, q_k, m_k)) \rangle. \end{aligned}$$

That is, the transition is chosen nondeterministically.

We claim that the succinct transition system has an acyclic plan if and only if the Turing machine accepts without violating the space bound.

If the Turing machine violates the space bound, then $h > e(n)$ and an accepting state cannot be reached because no further operator will be applicable.

Otherwise, it can be inductively shown that from a computation tree of an accepting ATM we can extract a conditional plan that always reaches a goal state, and vice versa.

A computation tree directly maps to a plan of the same form.

- (1) An accepting terminal configuration maps to a terminal plan node.
- (2) An \exists configuration maps to an operator node that applies the deterministic operator corresponding to the transition that is made.
- (3) A \forall configuration maps to an operator node that applies the nondeterministic operator corresponding to the all the transitions that are possible, followed by a branch node that branches maximally, that is, under all observable state variables, which indicate which of the alternative transitions actually took place.

Mapping from plans to computation trees is similar to the mapping in Theorem 19 except that the states encountered cannot be directly mapped to ATM configurations as the former do not contain the contents of the tape of the ATM.

So, because all alternating Turing machines with an exponential space bound can be in polynomial time translated into a nondeterministic planning problem with partial observability, all decision problems in AEXPSPACE are polynomial time many-one reducible to nondeterministic planning, and the plan existence problem is AEXPSPACE-hard and consequently 2-EXP-hard. \square

Vardi and Stockmeyer [1985] establish the 2-EXP-hardness of the satisfiability of the branching time temporal logic CTL* by a simulation of exponential-time alternating Turing machines, but their proof idea is different from ours. Their proof encodes ATM configurations as sequences of states, each corresponding to one tape cell, and testing the correctness of the differences between two consecutive tape cells by CTL* formulae. Our proof encodes the ATM configurations in the belief state – implicitly represented by a plan – by utilizing partial observability.

When operators are restricted to be deterministic or state-independent, complexity of the plan existence problem in the most general cases decreases.

Our proofs for the EXP-hardness and 2-EXP-hardness of plan existence for planning with full and partial observability use nondeterminism. Also the EXP-hardness proof of Littman [1997] uses nondeterminism. The question arises whether complexity is lower when all operators are deterministic. This turns out to be the case for planning with full (Theorem 24) and partial observability (Theorem 31), but without observability there is no difference: the EXPSPACE-hardness proof in that case uses deterministic operators only. In the partially observable case determinism guarantees that the size of the current belief state can never increase, and this leads to reduced complexity.

Interestingly, deterministic planning with one initial state is exactly as complex both with and without conditional effects, but for planning with restrictions on observability there is a difference in computational complexity. With deterministic state-independent operators the uncertainty about actual current state monotonically decreases and the values of state variables may never change without the new value being known, leading to a belief space with a particularly simple structure.

4.3 Plans with loops

The complexity results of the above plan existence problems hold also when loops are allowed in the plans. Loops are needed for finitely representing repetitive strategies that

do not have an upper bound on execution length. A typical problem would be to toss a die until its value is six. Assuming that the probability of the die falling on every side is non-zero, eventually getting six has probability 1, although the zero-probability event of unsuccessfully throwing the die infinitely many times is still possible.

In this section we discuss how the earlier results are extended to the case with loops in the plans. Looping is not possible in the unobservable case as it is impossible to decide when to stop looping, but for the fully and partially observable cases looping is applicable.

The problem looping plans cause for the proofs of Theorems 19 and 21 is that a Turing machine computation of infinite length is not accepting but the corresponding infinite length zero-probability plan execution is allowed to be a part of a plan. Hence for acyclic plans there is this mismatch between plans and accepting computation trees.

To eliminate infinite plan executions we have to modify the Turing machine simulations given in the proofs of those theorems. The modification involves counting the length of the plan executions and rejecting when the plan has been executed so far that at least one configuration has been visited more than once. This modification makes looping plans ineffective, and in the absence of loops the simulation is faithful.

Theorem 22 *The problem of testing the existence of an acyclic plan for problem instances with full observability (PLANSAT-FO-ND-RG) is EXP-hard, even with the restriction to only one initial state.*

PROOF. This is an easy modification of the proof of Theorem 19. If there are n state variables, an acyclic plan exists if and only a plan with execution length at most 2^n exists, because visiting any state more than once is unnecessary. Plans that rely on loops can be invalidated by counting the number of actions taken and failing when this exceeds 2^n . This counting can be done by having $n+1$ auxiliary state variables c_0, \dots, c_n that are initialized to false. Every operator $\langle p, e \rangle$ is extended to $\langle p, e \wedge t \rangle$ where t is an effect that increments the binary number encoded by c_0, \dots, c_n by one until the most significant bit c_n becomes one. The goal G is replaced by $G \wedge \neg c_n$.

Therefore, a plan exists if and only if an acyclic plan exists if and only if the alternating Turing machine accepts. \square

For the fully observable case counting the execution length does not pose a problem because we only have to count an exponential number of execution steps, which can be represented by a polynomial number of state variables, but in the partially observable case we need to count a doubly exponential number of execution steps, as the number of belief states to be visited may be doubly exponential. A binary representation of these numbers requires an exponential number of bits, and we cannot use an exponential number of state variables for representing them, because the reduction to planning would not be polynomial time. However, partial observability with only a polynomial number of auxiliary state variables has the power to count doubly exponentially far.

Theorem 23 *The problem of testing the existence of an acyclic plan for problem instances with partial observability (PLANSAT-PO-ND-RG) is 2-EXP-hard. This holds even when there is only one initial state.*

PROOF. We extend the proof of Theorem 21 by a counting scheme that makes cyclic plans ineffective. We show how counting transitions can be achieved within a succinct

transition system obtained from the alternating Turing machine and the input string in polynomial time.

Because representing the number of transitions as a binary number requires an exponential number of state variables, we cannot represent the number explicitly. Instead, we use a randomizing technique for forcing the plan to implicitly count the number of Turing machine transitions so far. The technique has some resemblance to the idea in simulating exponentially long working tapes in proofs of Theorems 19 and 21, but now the idea is applied to incrementing a number that has exponentially many digits.

For a succinct transition system with n state variables (representing the Turing machine configurations) executions that visit each belief state at most once may have length 2^{2^n} . Representing these numbers requires 2^n binary digits. We introduce $n + 1$ new unobservable state variables d_0, \dots, d_n for representing the index of *one of the digits* and v_d for the value of that digit, and new state variables c_0, \dots, c_n through which the plan indicates changes to the count of Turing machine transitions. There is a set of operators by means of which the plan sets the values of these variables before applying an operator corresponding to a Turing machine transition.

The idea of the construction is the following. Whenever the count of TM transitions is incremented, one of the 2^n digits in the count changes from 0 to 1 and all of the less significant digits change from 1 to 0. The plan is forced to communicate the index of the digit that changes from 0 to 1 by the state variables c_0, \dots, c_n . The unobservable state variables d_0, \dots, d_n, v_d store the index and value of one of the digits, that we call *the watched digit*, and they are used for checking that the reporting of c_0, \dots, c_n by the plan is truthful. On each execution only one digit can be the watched one and tested for correctness, but this suffices to invalidate plans that incorrectly report the increments, as a valid plan has to reach the goals on every possible execution and every digit will be the watched one on one of the executions. Plan execution fails when reporting is false or when the count exceeds 2^{2^n} . For this reason a plan for the succinct transition system exists if and only if an acyclic plan exists if and only if the Turing machine accepts the input string.

Next we exactly define how the succinct transition systems defined in the proof of Theorem 21 are extended to prevent unbounded looping in plans by means of a counter.

The initial state description is extended with the conjunct d_v to signify that the watched digit is initially 0 (all the digits in the counter implicitly represented in the belief state are 0.) The state variables d_0, \dots, d_n may have any values which means that the watched digit is chosen randomly. The state variables $d_v, d_0, \dots, d_n, d_v$ are all unobservable so that the plan does not know which digit is watched.

There is also a failure flag f that is initially set to false by having $\neg f$ in the initial states formula.

The goal G is extended to $G \wedge \neg f \wedge ((d_0 \wedge \dots \wedge d_n) \rightarrow \neg d_v)$ to prevent executions that lead to setting f true or that have length exceeding 2^{2^n} .

Then we extend the operators simulating the Turing machine transitions, as well as introduce new operators for the plan to indicate which digit changes from 0 to 1.

The operators for indicating the digit that changes are

$$\begin{aligned} \langle \top, c_i \rangle & \text{ for all } i \in \{0, \dots, n\} \\ \langle \top, \neg c_i \rangle & \text{ for all } i \in \{0, \dots, n\} \end{aligned}$$

The operators for Turing machine transitions are extended with the randomized test that the digit the plan claims to change from 0 to 1 is indeed the one: every operator $\langle p, e \rangle$ defined

in the proof of Theorem 21 is replaced by $\langle p, e \wedge t \rangle$ where the test t is the conjunction of the following effects.

$$\begin{aligned} ((c = d) \wedge d_v) &\triangleright f \\ (c = d) &\triangleright d_v \\ ((c > d) \wedge \neg d_v) &\triangleright f \\ (c > d) &\triangleright \neg d_v \end{aligned}$$

Here $c = d$ denotes $(c_0 \leftrightarrow d_0) \wedge \dots \wedge (c_n \leftrightarrow d_n)$ and $c > d$ encodes the greater-than test for the binary numbers encoded by c_0, \dots, c_n and d_0, \dots, d_n .

The above effects do the following.

- (1) When the plan claims that the watched digit changes from 0 to 1 and the value of d_v is 1, fail.
- (2) When the plan claims that the watched digit changes from 0 to 1, change d_v to 1.
- (3) When the plan claims that a more significant digit changes from 0 to 1 and the value of d_v is 0, fail.
- (4) When the plan claims that a more significant digit changes from 0 to 1, set the value of d_v to 0.

That these effects guarantee the invalidity of a plan that relies on unbounded looping is because the failure flag f will be set if the plan lies about the count, or the most significant bit with index $2^{n+1} - 1$ will be set if the count reaches $2^{2^{n+1} - 1}$. Attempts of unfair counting are recognized and consequently f is set to true because of the following.

Assume that the binary digit at index i changes from 0 to 1 (and therefore all less significant digits change from 1 to 0) and the plan incorrectly claims that it is the digit j that changes, and this is the first time on that execution that the plan lies (hence the value of d_v is the true value of the watched digit.)

If $j > i$, then i could be the watched digit (and hence $c > d$), and for j to change from 0 to 1 the less significant bit i should be 1, but we would know that it is not because d_v is false. Consequently on this plan execution the failure flag f would be set.

If $j < i$, then j could be the watched digit (and hence $c = d$), and the value of d_v would indicate that the current value of digit j is 1, not 0. Consequently on this plan execution the failure flag f would be set.

So, if the plan does not correctly report the digit that changes from 0 to 1, then the plan is not valid. Hence any valid plan correctly counts the execution length which cannot exceed $2^{2^{n+1} - 1}$. \square

5. COMPLEXITY OF PLANNING FOR MAINTENANCE

Complexity upper bounds for maintenance are obtained from the same bounds for repeated reachability in Section 6.

The lower bounds we established in the previous sections for the reachability objective also hold for the maintenance objective. The proofs are obtained from the proofs of Theorems 19, 20 and 21 by using a counter for the number of actions taken: before the counter reaches 2^n (for full observability) or 2^{2^n} (for no and partial observability) any state is a goal state, and after reaching the limit only the original goal states are goal states. Indefinitely staying in a goal state is made possible by a dummy operator that has no effects. This way the maintenance objective is satisfied if and only if the Turing machine accepts.

6. COMPLEXITY OF PLANNING FOR REPEATED REACHABILITY: UPPER BOUNDS

The repeated goal reachability criterion does not trivially reduce to the goal reachability criterion in the more complex cases with partial observability and nondeterminism, and we must derive algorithms specifically for this more general objective.

By Theorem 16 we directly obtain from hardness proofs of reachability goals also hardness for repeated reachability, and the algorithms we give for repeated reachability establish membership in complexity classes also for reachability goals.

It turns out that the complexities of all the three plan objectives coincide, and we just have to show membership in all the classes for which we gave hardness proofs in Section 4.

6.1 Planning with full observability

In the simplest deterministic fully observable cases the plan existence problem for repeated reachability can be solved by identifying paths and cycles in the transition graph.

Theorem 24 *Testing plan existence for succinct transition systems with deterministic operators and only one initial state (PLANSAT-??-D-RRG-1) and for deterministic fully observable succinct transition systems with several initial states (PLANSAT-FO-D-RRG) under the repeated reachability criterion is in PSPACE.*

PROOF. Let $\Pi = \langle A, I, O, G, A \rangle$ be a succinct transition system so that all operators in O are deterministic.

The fully observable problem with several initial states easily reduces to the one initial state case: just perform an iteration over all possible initial states, and test existence of a plan separately for each. Full observability is essential for this reduction to be correct because branching so that a plan for every initial state can be constructed separately is otherwise in general not possible.

Let s_I be the initial state. The plan existence problem can be solved by in polynomial space as follows. Iterate over all states to find s such that

- (1) $s \models G$,
- (2) s is reachable from s_I by at most $2^{|A|}$ steps, tested by $\text{FOreach}(A, O, s_I, s, |A|)$, and
- (3) there is a non-empty path from s to itself. This is by testing $\text{FOreach}(A, O, s', s, |A|)$ for every $o \in O$ with $\text{img}_o(s) = \{s'\}$. If one of these calls returns true, a non-empty path from s to itself exists.

The reachability tests are for paths of length at most $2^{|A|}$ because the number of states is $2^{|A|}$ and if a path exists then a path of at most length $2^{|A|} - 1$ exists.

Iteration over all states s can be done in polynomial space, and the procedure calls $\text{FOreach}(A, O, s, s', m)$ need polynomial space in m and the sizes of the inputs. The procedure FOreach is given in Figure 2. \square

Planning with full observability has properties that allow using a simpler definition of plans. Essentially, plans with several nodes are unnecessary, as no memory about the prior steps in the plan execution need to be retained. Hence plans can be viewed as mappings from states to actions, or equivalently, each state can be taken as a plan node, and after execution the associated action, the plan always branches for every successor state and continues execution from the node corresponding to the successor state. In connection with Markov decision processes [Howard 1960; Puterman 1994] this kind of history-independent policies have been called *stationary*.

```

procedure FOreach( $A, O, s, s', m$ )
if  $m \leq 0$  then
  if ( $s = s'$  or there is  $o \in O$  such that  $\{s'\} = \text{img}_o(s)$ ) then return true
  else return false
else
  for each valuation  $s''$  of  $A$  do
    if FOreach( $A, O, s, s'', m - 1$ ) and FOreach( $A, O, s'', s', m - 1$ ) then return true
  end
return false

```

Fig. 2. Algorithm for finding a path of length $\leq 2^m$ from s to s'

Theorem 25 *Assume there is a plan π solving a succinct transition system $\Pi = \langle A, I, O, G, A \rangle$ under the RRG criterion. Let S be the set of valuations of A . Then there is a function $x : S \rightarrow O$ and a plan $\pi' = \langle S, b, l \rangle$ also solving Π such that*

- (1) $b = \{\langle U(s), s \rangle \mid s \in S, s \models I\}$, and
- (2) $l(s) = \langle x(s), \{\langle U(s'), s' \rangle \mid s' \in \text{img}_{x(s)}(s)\} \rangle$,

where $U(s) = \bigwedge (\{a \in A \mid s \models a\} \cup \{\neg a \mid a \in A, s \models \neg a\})$, and its execution graph is $\langle M, E \rangle$ where

- (1) $M = S \times S$
- (2) $E = \{(\langle s, s \rangle, \langle s', s' \rangle) \in M \times M \mid o \in O, s' \in \text{img}_{x(s)}(s)\}$.

PROOF. We construct from the plan π and its execution graph the plan π' with the execution graph $\langle M, E \rangle$.

- (1) First transform π by making branching more fine-grained so that for every plan node n there is only one state s so that $\langle s, n \rangle$ is reachable from the initial nodes in the execution graph.

Let n be a plan node such that there are for states s and s' such that $s \neq s'$ nodes $\langle s, n \rangle$ and $\langle s', n \rangle$ in the execution graph. Now for every state s

- (a) rename nodes $\langle s, n \rangle$ to $\langle s, n_s \rangle$ (also in the edges),
- (b) add node n_s to the plan with $l(n_s) = l(n)$,

For all plan nodes n' such that for $l(n') = \langle o, B \rangle$ there is $\langle \phi, n \rangle \in B$ and for $\langle \phi, n \rangle \in B$ replace $\langle \phi, n \rangle$ in B by $\langle U(s), n_s \rangle$ for all states s such that $s \models \phi$.

Finally remove node n from the plan.

The above modifications change the structure of the plan but only rename nodes in the execution graph. Hence their do not affect the satisfaction of the RRG or other criteria.

- (2) Now there may be several plan nodes with the same state s . If there is node n_s so that there is a path in the execution graph from $\langle s, n_s \rangle$ to a node in $G \times N$ that does not go through a node $\langle s, n'_s \rangle$ and $\langle s, n_s \rangle$ is reachable from one of the nodes $\langle s_I, m \rangle$ such that $s_I \models I$ and there is $\langle \phi, m \rangle \in b$ such that $s_I \models \phi$, then replace all edges to n'_s everywhere in the plan by edges to n_s , delete the now unreachable plan node n'_s , and modify the execution graph accordingly by replacing all edges to $\langle s, n'_s \rangle$ by edges to $\langle s, n_s \rangle$ and delete the nodes $\langle s, n'_s \rangle$.

If there is no such node n_s then it means that state s cannot be reached on any plan execution. Hence we may replace all edges in the plan nodes n'_s by edges to one arbitrarily chosen n_s .

We can show that after each of these steps the plan still satisfies the RRG criterion.

```

procedure prune( $S, O, G$ );
 $W_{-1} := S$ ;
 $W_0 := \emptyset$ ;
repeat
   $W'_0 := W_0$ ;
   $W_0 := W'_0 \cup \bigcup_{o \in O} \text{wpreimg}_o(W'_0 \cup G)$ ;
until  $W_0 = W'_0$ ; (* States from which a goal state can be reached by  $\geq 1$  operators *)
 $i := 0$ ;
repeat
   $i := i + 1$ ;
   $k := 0$ ;
   $S_0 := \emptyset$ ;
  repeat
     $k := k + 1$ ; (* States from which a state in  $G$  is reachable with  $\leq k$  steps. *)
     $S_k := S_{k-1} \cup \bigcup_{o \in O} (\text{wpreimg}_o(S_{k-1} \cup G) \cap \text{spreimg}_o(W_{i-1} \cup G))$ ;
  until  $S_k = S_{k-1}$ ; (* States that stay within  $W_{i-1}$  before reaching  $G$ . *)
   $W_i := S_k$ ;
until  $W_i = W_{i-1}$ ; (* States in  $W_i$  stay within  $W_i$  before reaching  $G$ . *)
return  $W_i$ ;

```

Fig. 3. Algorithm for detecting a loop that eventually makes progress

- (3) After repeatedly doing the previous step, there will be for any state s only one node n . Now we can identify these nodes n with these states, and the execution graph has the desired form.

□

So in the fully observable case plans can be equivalently represented as mappings from states to actions. This is the representation of plans we use in establishing the complexity of the most general fully observable problem.

For nondeterministic operators, full observability and repeated reachability goals we give a new algorithm. The algorithm solves a problem that generalizes the problem solved by an algorithm by Cimatti et al. [2003] (the global strong cyclic algorithm) for our simplest objective of reachability goals. The algorithm runs in polynomial time in the size of the state space and hence yields an exponential time upper bound for the plan existence problem. The algorithm uses the subprocedure *prune* given in Figure 3.

We introduce some terminology. Let S be a set of states, O a set of operators, and $x : S \rightarrow O$ a mapping from states to operators. A sequence s_0, \dots, s_n of states is an *execution* if for all $i \in \{1, \dots, n\}$ there is $o \in O$ such that $s_i \in \text{img}_o(s_{i-1})$, and it is an *execution of x* if $s_i \in \text{img}_{x(s_{i-1})}(s_{i-1})$ for all $i \in \{1, \dots, n\}$.

Lemma 26 (Procedure prune) *Let S be a set of states, O a set of operators and $G \subseteq S$ a set of states. Then the procedure call $\text{prune}(S, O, G)$ will terminate after a finite number of steps returning a set of states $W \subseteq S$ such that there is function $x : W \rightarrow O$ such that*

- (1) *for every $s \in W$ there is an execution s_0, s_1, \dots, s_n of x with $n \geq 1$ such that $s = s_0$ and $s_n \in G$,*
- (2) *$\text{img}_{x(s)}(\{s\}) \subseteq W \cup G$ for every $s \in W$, and*
- (3) *W is the maximal subset of S having a function x satisfying the above properties. More precisely, for any state not in W and for any plan, after some finite number of execution steps according to that plan a state may be reached from which no goal state can be reached by any sequence of actions.*

Formally, for every $s \in S \setminus W$ and function $x' : S \rightarrow O$ there is an execution s_0, \dots, s_n of x' such that $s = s_0$ and there is no $m \geq n$ and execution s_n, s_{n+1}, \dots, s_m such that $s_m \in G$.

PROOF. The proof is by two nested induction proofs each corresponding to a repeat-until loop in the procedure *prune*. If there is no plan that is guaranteed to reach a goal state from a state s , then this is because for any plan after some number of executions steps i it is possible to reach a state from which no sequence actions can reach a goal state. A plan covering all other states exists with an execution reaching a goal state in some h steps. The outer loop and induction go through $i = 0, 1, 2, \dots$ and the inner loop and induction through $h = 0, 1, 2, \dots$

Induction hypothesis: There is function $x : W_i \rightarrow O$ such that

- (1) for every $s \in W_i$ there is an execution s_0, \dots, s_n of x such that $n \geq 1$, $s = s_0$ and $s_n \in G$,
- (2) $\text{img}_{x(s)}(\{s\}) \subseteq W_{i-1} \cup G$ for every $s \in W_i$, and
- (3) for all functions $x' : S \rightarrow O$ and states $s \in S \setminus W_i$ there is $i' \in \{0, \dots, i\}$ and an execution $s_0, \dots, s_{i'}$ of x' such that $s_0 = s$ and there is no $h \geq i'$ and execution $s_{i'}, s_{i'+1}, \dots, s_h$ such that $s_h \in G$.

Base case $i = 0$:

- (1) W_0 has been computed to fulfill exactly this property.
- (2) Because $W_{-1} = S$ trivially $\text{img}_{x(s)}(\{s\}) \subseteq W_{-1} \cup G$.
- (3) States $s \in W_0 \setminus W_{-1}$ are exactly those states in which no operator sequence leads to G .

Inductive case $i \geq 1$: For the inner *repeat-until* loop we prove inductively the following.

Induction hypothesis: There is function $x : S_k \rightarrow O$ such that

- (1) for every $s \in S_k$ there is an execution s_0, s_1, \dots, s_n of x such that $n \in \{1, \dots, k\}$, $s = s_0$ and $s_n \in G$,
- (2) $\text{img}_{x(s)}(\{s\}) \subseteq W_{i-1} \cup G$ for every $s \in S_k$, and
- (3) for all functions $x' : S \rightarrow O$ and states $s \in S \setminus S_k$ either
 - (a) there is $i' \in \{0, \dots, i\}$ and an execution $s_0, \dots, s_{i'}$ of x' such that $s_0 = s$ and there is no $h \geq i'$ and execution $s_{i'}, s_{i'+1}, \dots, s_h$ such that $s_h \in G$, or
 - (b) there is no $k' \in \{1, \dots, k\}$ and an execution $s_0, \dots, s_{k'}$ of x' such that $s_0 = s$ and $s_{k'} \in G$.

Base case $k = 0$: Because $S_0 = \emptyset$, cases (1) and (2) trivially hold for every $s \in S_0$. It remains to show the third component of the induction hypothesis.

- (3) For any $s \in S \setminus S_0 = S$ (3b) is satisfied because it requires executions to be longer than $k = 0$.

Inductive case $k \geq 1$: We extend the function $x : S_{k-1} \rightarrow O$ to cover states in $S_k \setminus S_{k-1}$. Let s be any state in S_k . If $s \in S_{k-1}$ then properties (1) and (2) are by the induction hypothesis. Otherwise $s \in S_k \setminus S_{k-1}$. Therefore by definition of S_k , $s \in \text{wpreimg}_o(S_{k-1} \cup G) \cap \text{spreimg}_o(W_{i-1} \cup G)$ for some $o \in O$.

- (1) Because $s \in \text{wpreimg}_o(S_{k-1} \cup G)$ for some $o \in O$, either $s \in \text{wpreimg}_o(S_{k-1})$ or $s \in \text{wpreimg}_o(G)$.
 If $s \in \text{wpreimg}_o(G)$ then we set $x(s) = o$. The desired execution just consists of s and a state $s' \in G$.
 If $s \in \text{wpreimg}_o(S_{k-1}) \setminus \text{wpreimg}_o(G)$ then there is a state $s' \in S_{k-1}$ such that $s' \in \text{img}_o(\{s\})$. By the induction hypothesis there is an execution of x starting from s' that ends in a goal state. For s such an execution is obtained by prefixing with o , so we define $x(s) = o$.
- (2) Because $s \in \text{spreimg}_o(W_{i-1} \cup G)$, by (2) and (3) of Lemma 3 $\text{img}_o(\{s\}) \subseteq W_{i-1} \cup G$.
- (3) Take any $s \in S \setminus S_k$. Now for every operator $o \in O$, either $s \notin \text{spreimg}_o(W_{i-1} \cup G)$ or $s \notin \text{wpreimg}_o(S_{k-1} \cup G)$. Consider any function $x' : S \rightarrow O$ such that $x'(s) = o$. In the first case by the outer induction hypothesis there is $i' \in \{0, \dots, i-1\}$ and an execution $s_0, \dots, s_{i'}$ of x' such that $s_0 \in \text{img}_o(s)$ and there is no $h \geq i'$ and execution $s_{i'}, s_{i'+1}, \dots, s_h$ such that $s_h \in G$. Hence executing o first could similarly lead to the state $s_{i'}$ from which no goal could be reached, now requiring i steps.
 In the second case by the inner induction hypothesis for all $s' \in \text{img}_o(s)$ there is no execution of length $k-1$ ending in a goal state.
 Because this holds for any $o \in O$, every x' has one of these properties.

This completes the inner induction. To establish the induction step of the outer induction consider the following. The inner repeat-until loops ends when $S_k = S_{k-1}$. This means that $S_z = S_k$ for all $z \geq k$. Hence there is no upper bound k on the length of executions for reaching a goal state for (1) and (3). The outer induction hypothesis is obtained from the inner induction hypothesis by removing the upper bound and replacing S_k by W_i . By construction $W_i = S_k$.

This finishes the outer induction proof. The claim of the lemma is obtained from the outer induction hypothesis by noticing that the outer loop exits when $W_i = W_{i-1}$ (it will exit after a finite number of iterations because W_0 is finite and its size decreases on every iteration) and then we can replace both W_i and W_{i-1} by W to obtain the claim of the lemma. \square

Theorem 27 *Testing plan existence for succinct transition systems with full observability under the repeated reachability criterion (PLANSAT-FO-ND-RRG) is in EXP.*

PROOF. The main procedure of the decision procedure is in Figure 4. Given a succinct transition system Π , we can produce the corresponding transition system $F(\Pi) = (S, I, O, G, P)$ in exponential time. Then we call the procedure `planexistsFO(S, I, O, G)`. The procedure first sets $G_{ne} := G$, and then repeatedly eliminates from G_{ne} those goal states from which reaching a goal state by a plan with a non-empty execution cannot be guaranteed. The elimination uses the subprocedure *prune* (Figure 3 and Lemma 26.)

In the end G_{ne} will be the maximal set of goal states from which reaching a goal state in G_{ne} is guaranteed. The number of iterations is bounded by the number of states, and each iteration has a runtime polynomial in the number of states. Hence the total runtime of this stage is exponential in the size of the succinct transition system.

Finally, we test whether from the initial state we can always reach a goal state (one of the remaining). Now, there is a plan x so that from any state that can be reached from an

```

procedure planexistsFO( $S, I, O, G$ )
 $G_{ne} := G$ ;
repeat
   $W := \text{prune}(S, O, G_{ne})$ ;
   $G'_{ne} := G_{ne}$ ;
   $G_{ne} := G_{ne} \cap W$ ;
until  $G_{ne} = G'_{ne}$ ;
if  $I \subseteq W$  then return true else return false;

```

Fig. 4. Algorithm for nondeterministic planning with full observability

initial state by using this plan, there is a path to a goal state in G_{ne} , including the state in G_{ne} . Hence the RRG criterion is satisfied.

- (1) There is a plan such that for all states in W a state in G_{ne} is reached.
- (2) There is no plan for states not in W .

If there is a plan, the procedure returns yes.

If the procedure returns yes, there is a plan. \square

6.2 Planning with unobservability

Next we investigate the repeated reachability criterion for planning without observability and for the most general case of partial observability. As pointed out in Example 12, the repeated reachability criterion does not require that at any point of time it is known that the current state is one of the goal states. This is why planning with the repeated reachability criterion does not trivially reduce to planning with goal reachability.

Representing and recognizing dependencies between a belief state and its predecessor belief states becomes necessary, and unlike for reachability goals, plans cannot be formalized as mappings from the current belief state to an action, that is, the belief states do not include all the necessary information needed for testing the satisfaction of the plan objective.

The EXPSPACE lower bound for the complexity of the most general unobservable problem PLANSAT-UO-ND-RRG is given by Theorem 20. We prove the EXPSPACE upper bound by devising a corresponding decision procedure. The structure of plans for unobservable problems is the same as for the PSPACE-complete fully observable deterministic problem with one initial state considered in Theorem 24: a sequence of actions followed by a loop. The complication in the unobservable case is that it is not necessarily ever known when a goal state is visited. The algorithm we give is an extension of the algorithm given in Theorem 24. First a path to the starting belief state of a loop is found. Then a loop, that is a non-trivial path from the belief state to itself, is found. Simultaneously with the loop we find for each state in the belief state an execution that visits a goal state, showing that the loop satisfies the RRG criterion. All this can be done by using only an exponential amount of memory, yielding the EXPSPACE membership of the decision problem.

To show that this approach is complete (finds a plan whenever one exist), we first show that whenever a plan exists, a plan with the given structure exists, that is, repeats a loop that maps a given belief state to itself. Then we derive an upper bound on the length of such a loop.

Let o_1, \dots, o_n be a loop satisfying the RRG criterion. In the unobservable case plans with an infinite execution reduce to a particularly simple form: a sequence σ of operators (the prefix) followed by another sequence σ' of operators (the loop) that is repeated to

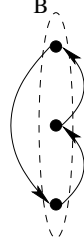


Fig. 5.

produce the infinite sequence of operators $\sigma\sigma'\sigma'\sigma'\dots$. This is the structure of infinite paths in a graph with only nodes of degree one. This plan induces an infinite sequence of belief states, similarly consisting of a prefix σ_B followed by a loop σ'_B , but the lengths of σ_B and σ'_B do not necessarily equal the lengths of σ and σ' .

Any plan can be transformed so that the length of the prefix and the loop for both operators and the belief states coincide. Let B_1 be the belief state reached after executing σ . After also executing σ' we may reach some other belief state $B_2 \neq B_1$. Let $B_1, B_2, \dots, B_n, \dots$ be the sequence of belief states obtained this way, each B_i reached from the initial belief state by executing σ and then σ' $i - 1$ times. Because the number of belief states is finite, for some $n \geq 1$ and $k \in \{1, \dots, n - 1\}$, $B_k = B_n$. Let n be the minimal such n . Now from B_n executing σ' $n - k$ times takes us to B_n . Now we can view $\sigma\sigma'^{k-1}$ as the prefix of the plan and σ'^{n-k} as the loop. This loop has the property that B_n is its first belief state on every iteration.

We transform the plan further, so that visiting a goal state in G is possible (considering nondeterminism) when executing the loop only once starting from any $s \in B_n$. Notice that in general for some states $s \in B_n$ there is no execution path leading to a goal state in only one execution of the loop: in the example in Figure 5 the loop consists of one operator only, the one depicted as arrows, only one of the three states is a goal state, and a goal state is visited on every third execution of the loop.

So consider a loop o_1, \dots, o_n with $n \geq 1$ and the corresponding belief states B_0, \dots, B_n where $B_0 = B_n$. Because the RRG criterion is satisfied $G \cap B_i \neq \emptyset$ for some $i \in \{0, \dots, n\}$. Hence for some $s \in B_0$ $img_{o_1, \dots, o_i}(s) \cap G \neq \emptyset$. Now for states $s' \in B_0$ such that $s \in img_{o_1, \dots, o_n}(s')$ visiting a state in G is possible with at most 2 iterations of the loop. Repeating this argument shows for every state in B_0 it is possible to visit a goal state by executing the loop at most $|B_0|$ times. Hence this loop can be concatenated $|B_0|$ times and we obtain a loop by executing which once for any state in B_0 visiting a goal state is possible.

We now derive an upper bound on the length of such a loop. Let o_1, \dots, o_n be a loop with belief states B_0, \dots, B_n such that $B_0 = B_n$ and an execution s_0, \dots, s_n visiting a goal state for *only one* of the states $s_0 \in B_0$, that is, $s_i \in G$ for some $i \in \{0, \dots, n\}$. Assume for some $i, j, k \in \{0, \dots, n\}$ such that $i \neq j \neq k \neq i$ both $B_i = B_j = B_k$ and $s_i = s_j = s_k$. If for $s_g \in G$ for $g \in \{i, \dots, j - 1\}$ then we obtain a shorter loop visiting G for s_0 by eliminating operators o_j, \dots, o_{k-1} , and if $g \in \{j, \dots, k - 1\}$ we eliminate operators o_i, \dots, o_{j-1} . Hence there need be at most two occurrences of any B_i, s_i , and the loop need have length at most $2^{|S|}2^{|S|}$. Now we can concatenate the loops for all $s \in B_0$,

```

procedure UOreach( $S, O, B, B', m$ )
if  $m \leq 0$  then
  if  $B = B'$  or (there is  $o \in O$  such that  $B \subseteq \text{prec}(o)$  and  $B' = \text{img}_o(B)$ )
    then return true
  else
    for each  $B'' \subseteq S$  do
      if UOreach( $S, O, B, B'', m - 1$ ) and UOreach( $S, O, B'', B', m - 1$ ) then return true;
    end;
  return false

```

Fig. 6. Algorithm for finding a sequence of actions leading from B to B'

```

procedure UOreachr( $S, O, B, r, B', r', m, Q$ )
if  $m \leq 0$  then
  if ( $B = B'$  and  $r = r'$ )
    or (there is  $o \in O$  such that  $B \subseteq \text{prec}(o)$  and  $B' = \text{img}_o(B)$ 
      and  $r'(q) \in \text{img}_o(r(q))$  or ( $r(q) = r'(q)$  and  $r(q) \in G$ ) for all  $q \in Q$ )
    then return true
  else
    for each  $B'' \subseteq S$  and function  $r'' : Q \rightarrow S$  do
      if UOreachr( $S, O, B, r, B'', r'', m - 1, Q$ ) and UOreachr( $S, O, B'', r'', B', r', m - 1, Q$ ) then return true;
    end;
  return false

```

Fig. 7. Algorithm for finding a sequence of action from B and B' that may visit G

obtaining a loop that can visit a goal state from any state in B_0 and having length at most $2^{|S|} 2^{|S|^2}$.

$$\log_2(2^n 2^{n^2}) = 1 + \log_2 2^n + \log_2 n^2 = 1 + n + 2 \log_2 n$$

Theorem 28 *Testing plan existence for succinct transition systems without observability under the repeated reachability criterion (PLANSAT-UO-ND-RRG) is in EXPSPACE.*

PROOF. Given a succinct transition system Π , we first produce the corresponding transition system $F(\Pi) = \langle S, I, O, G, (S) \rangle$ in exponential time. Then we find a plan by using the algorithm with the main procedure given in Figure 8. The outer loop iterates over all belief states B that may be the first in the loop that follows. This iteration takes only exponential space.

Let $n = |S|$ be the cardinality of the state space.

For each B test with UOreach($O, I, B, 2^n$) that B is reachable from I . If it is, iterate over all functions $r' : B \rightarrow G$ and test whether for some $o \in O$ $\text{img}_o(s) \neq \emptyset$ for all $s \in B$ (o is applicable in all states in B) and UOreachr($S, O, \text{img}_o(B), r, B, r', n + n \lceil \log_2(|B|) \rceil$) returns true, that is, whether there is a non-empty sequence of actions that reaches B from B itself.

Because the search for the path takes place in the combined space of all belief states and all mappings from B to S , the cardinality of which is $2^n n^n$, an upper bound on the path length is similarly $2^n n^n$. For binary search in UOreachr this yields the upper bound

$$\log_2(2^n n^n) = \log_2 2^n + \log_2 [(2^{\log_2 n})^n] = n + \log_2 2^{n \log_2 n} = n + n \log_2 n$$

for the recursion depth.

The procedure goes through all sequences of actions of length $2^n n^n$, and hence whenever a plan exists one of length $< 2^n n^n$ will be found. \square

```

procedure UOplanexistence( $S, I, O, G$ )
for each  $B \subseteq S$ , operator  $o \in O$  and functions  $r : B \rightarrow S$  and  $r' : B \rightarrow G$  do
    if  $r(s) \in \text{img}_o(s)$  for all  $s \in B$ 
        and UOreach( $S, O, I, B, n$ )
            and UOreach( $S, O, \text{img}_o(B), r, B, r', n + n \lceil \log_2 n \rceil, B$ ) then return true
end do;
return false

```

Fig. 8. A decision procedure for testing plan existence without observability

6.3 Planning with partial observability

A 2-EXP upper bound for partially observable problems under the maintenance and reachability criteria can be directly obtained by using the algorithms for the corresponding fully observable problems (Theorems 27): view belief states (sets of states) as states and view subsets of the set of goal states as goal states. For the most general repeated reachability criterion establishing membership in 2-EXP is not as easy.

Theorem 29 *Testing plan existence for succinct transition systems with partial observability under the repeated reachability criterion (PLANSAT-PO-ND-RRG) is in 2-EXP.*

PROOF. We use the algorithm for fully observable planning given in Theorem 27 as a basis of a decision procedure for PLANSAT-PO-ND-RRG. Instead of states we use belief states extended with information on executions, similarly to the proof of Theorem 28. The set of these extended belief states has size that is exponential in the number of states, and leads to 2-EXP membership instead of EXP membership.

Let S be the set of all states. Then an extended belief state is a pair $\langle B, B' \rangle$ where $B \subseteq C$ for some observational class C and $B' \subseteq B$.

We use extended belief states for finding belief states B such that from $\langle B, B' \rangle$ extended belief states $\langle B', \emptyset \rangle \in G_{ne}$ can be reached. This means that for every state in B on some execution a goal state can be reached.

To use the algorithm in Theorem 27 we define the functions $\text{spreimg}_o^E(W)$ and $\text{wpreimg}_o^E(W)$ for sets W of extended belief states and use them respectively instead of $\text{spreimg}_o(W)$ and $\text{wpreimg}_o(W)$.

For an extended belief state $\langle B, B' \rangle$ and a set W of extended belief states, $\langle B, B' \rangle \in \text{wpreimg}_o^E(W)$ if for every $s \in B'$ there is an observational class C and $\langle B'', B''' \rangle \in W$ such that

- (1) $B'' = \text{img}_o(B) \cap C$ and
- (2) either $\text{img}_o(s) \cap B''' \neq \emptyset$ or $\text{img}_o(s) \cap G \neq \emptyset$.

Notice that this definition does not require covering all observational classes that might result from B with o .

For an extended belief state $\langle B, B' \rangle$ and a set W of extended belief states, $\langle B, B' \rangle \in \text{spreimg}_o^E(W)$ if for every observational class C such that $\text{img}_o(B) \cap C \neq \emptyset$ there is $\langle B'', B''' \rangle \in W$ such that

- (1) $B'' = \text{img}_o(B) \cap C$.

The idea with the weak preimages of sets of extended belief states is that an extended belief state $\langle B, B' \rangle$ is in the weak preimage if for every state in it there is an execution that

```

procedure planexistsPO( $S, I, O, G, (C_1, \dots, C_n)$ )
 $G_{ne} := \{\langle B, \emptyset \rangle \mid i \in \{1, \dots, n\}, B \subseteq C_i\}$ ;
repeat
   $W := \text{prune}^E(S, O, G_{ne})$ ;
   $G'_{ne} := G_{ne}$ ;
   $G_{ne} := \{\langle B, \emptyset \rangle \in G'_{ne} \mid \langle B, B \rangle \in W\}$ ;
until  $G_{ne} = G'_{ne}$ ;
if for all  $i \in \{1, \dots, n\}, \langle I \cap C_i, \emptyset \rangle \in W$  then return true else return false;

```

Fig. 9. Algorithm for nondeterministic planning with partial observability

eventually leads to a goal state. In the function *prune* this is needed for guaranteeing the satisfaction of the RRG criterion.

For strong preimages we just want to guarantee that the execution does not take us out of the good belief states.

Notice that for fully observable problems, when the sets C of observational classes are singleton sets, there is an exact match between the functions $\text{spreimg}_o^E(W)$ and $\text{wpreimg}_o^E(W)$ respectively with $\text{spreimg}_o(W)$ and $\text{wpreimg}_o(W)$.

Define prune^E exactly like *prune* but by replacing $\text{spreimg}_o(\cdot)$ and $\text{wpreimg}_o(\cdot)$ respectively by $\text{spreimg}_o^E(\cdot)$ and $\text{wpreimg}_o^E(\cdot)$.

kesken \square

For deterministic operators the belief state size monotonically decreases as plan execution proceeds. This leads to a particularly simple structure for plans when executions are infinite: they consist of a branching non-looping part that ends in any of belief states B_1, \dots, B_n . The sum of the cardinalities of these belief states is less than or equal to the cardinality of the initial belief state. Further, for each belief state $B \in \{B_1, \dots, B_n\}$ there is a non-branching loop to itself so that for any starting state $s \in B$ there is an execution that visits a goal state, and all belief states in that loop have the same cardinality. Observations at this stage do not reduce the size of the belief states.

Theorem 30 *Assuming partial observability and deterministic operators, testing the existence of a plan that always ends in one of the belief states B_1, \dots, B_n can be done in exponential space.*

PROOF. The idea is similar to the EXPSpace membership proof of planning without observability: go through all possible intermediate stages of a plan by binary search (intermediate stage = a maximal set of plan nodes of which none is a successor of another.) Determinism yields an exponential upper bound on the sum of the cardinalities of the belief states that are possible after branching and a given number of actions, and it also entails that no belief state has to be visited more than once (= acyclic plans). Hence plan executions have doubly exponential length and binary search needs only exponential recursion depth. Because only an exponential amount of memory is needed at each call, the whole memory consumption is exponential.

Let $F(\Pi) = \langle S, I, O, G, (C_1, \dots, C_k) \rangle$ be the transition system corresponding to a given succinct transition system Π . For belief state B and set L of belief states with $\sum_{B' \in L} |B'| \leq |S|$, test reachability of belief states in L from B with plans of depth 2^i by algorithm in Figure 10.

We can now test plan existence by calling $\text{POreach}(S, O, B, L, (C_1, \dots, C_k), 2^m)$ for every $B = I \cap C_i, i \in \{1, \dots, k\}$. Here $L = \{B_1, \dots, B_n\}$ and m is the number of state

```

procedure POreach( $S, O, B, L, (C_1, \dots, C_k), i$ )
if  $i = 0$  then
  begin
    for each  $j \in \{1, \dots, k\}$ 
      if  $\text{img}_o(B) \cap C_j \subseteq B'$  for no  $o \in O$  such that  $B \subseteq \text{prec}(o)$  and  $B' \in L$ 
      and  $B \cap C_j \subseteq B'$  for no  $B' \in L$ 
      then return false
    end
    return true
  end
else
  for each  $L' \subseteq 2^S$  such that  $\sum_{B' \in L'} |B'| \leq |B|$  and
    for every  $B' \in L', B' \subseteq C_j$  for some  $j \in \{1, \dots, k\}$ 
    if POreach( $S, O, B, L', (C_1, \dots, C_k), i - 1$ )
      and POreach( $S, O, B'', L, (C_1, \dots, C_k), i - 1$ ) = true for all  $B'' \in L'$ 
      then return true
    end for
  return false

```

Fig. 10. Algorithm that tests reachability of sets of belief states by deterministic operators in EXPSPACE

```

procedure planexistsPOdet( $S, I, O, G, (C_1, \dots, C_k)$ )
 $n := |S|;$  (* Number of states *)
for all sets  $L \subseteq 2^S$  such that  $\sum_{B \in L} |B| < |I|$  do
  if POreach( $S, O, I, L, (C_1, \dots, C_k), n$ ) then
    if for every  $B \in L$  (* Starting belief state of a loop *)
      for some  $r : S \rightarrow 2^S$  such that  $B \subseteq \bigcup \{r(s) \mid s \in G\}$ 
        UOreach( $O, B, \{(s, s) \mid s \in B\}, B, r, n + \lceil n \log_2 n \rceil$ )
      then return true;
    end for
  return false

```

Fig. 11. Algorithm for testing plan existence for deterministic operators and partial observability

variables. The algorithm always terminates, and a plan exists if and only if answer *true* is obtained in all cases.

The space consumption is (only) exponential because the recursion depth is exponential and the sets $L' \subseteq 2^S$ with $\sum_{B' \in L'} |B'| \leq |B|$ have size $\leq |S|$. This small sets L' suffice because all operators are deterministic, and after any number of actions, independently of how the plan has branched, the sum of the cardinalities of the possible belief states is not higher than the cardinality of the set of initial states. This is because the size of a successor belief state cannot be bigger, and the sum of sizes of belief states following a branch equals the size of the predecessor belief state. \square

Theorem 31 *Testing plan existence for succinct transition systems with partial observability and deterministic operators under the repeated reachability criterion (PLANSAT-PO-D-RRG) is in EXPSPACE.*

PROOF. The procedure for testing plan existence is given in Figure 11. The algorithm loops over the sets of starting belief states of loops (the sum of their cardinalities is at most the cardinality of the initial belief state).

For each potential set of loops' starting belief states, the existence of an acyclic plan for reaching these belief states is tested by the procedure *POreach* that runs in exponential space by Theorem 30.

```

procedure notapplicable( $A, o, I, B$ )
  guess  $s : A \rightarrow \{T, F\}$  so that  $s \models I$ ;
  for each  $p \in A$  do
    if  $B(p) \neq U$  then  $s(p) := B(p)$ ;
  if  $s \not\models \text{prec}(o)$  then return true;
  return false

```

Fig. 12. A nondeterministic test for applicability of an operator in a 3-valued belief state

```

procedure 3app( $A, I, o, B, s, B', s'$ )
  if  $\text{img}_o(s) \neq \{s'\}$  then return false;
  if notapplicable( $A, o, I, B$ ) then return false;
  for each  $p \in A$  do
    if  $B'(p) \neq T$  and  $p$  in  $\text{eff}(o)$  then return false;
    if  $B'(p) \neq F$  and  $\neg p$  in  $\text{eff}(o)$  then return false;
    if  $B(p) \neq B'(p)$  and ( $p$  not in  $\text{eff}(o)$  and  $\neg p$  not in  $\text{eff}(o)$ ) then return false;
  end for
  return true

```

Fig. 13. Test for reachability of B', s' from B, s by 0 or 1 operators

Finally, the existence of a goal-visiting non-branching loop for each of the designated starting belief states of a loop is tested by the procedure *UOreachr*, which runs in exponential space by Theorem 28. \square

Planning is still simpler when operators are state-independent, that is, the changes made by the operator are always the same.

Theorem 32 *Testing plan existence for succinct transition systems with partial observability and deterministic state-independent operators under the repeated reachability criterion (PLANSAT-PO-ID-RRG) is in PSPACE.*

PROOF. Because the set of observable state variables is always the same and no information about the unobservable state variables can be obtained indirectly (through operators with conditional effects), it suffices to initially observe everything, branch, and for every resulting belief state find a sequential plan, consisting of a prefix and a loop, that visits a goal state, returns to the belief state, and then repeats the loop.

Testing the existence of such a loop is by the algorithm *3reach* in Figure 14.

The main procedure of the algorithm is given in Figure 16.

In these procedures belief states are represented as 3-valued valuations of state variables, that is, as functions $B : A \rightarrow \{T, F, U\}$. The values T and F respectively mean that the state variable has been assigned or observed to be *true* or *false*. The value U means that the value is the same as in the unknown initial state, and hence has not been assigned nor observed.

The notation $s \in B$ for 3-valued belief state $B : A \rightarrow \{T, F, U\}$, a fixed initial belief state I , and state $s : A \rightarrow \{T, F\}$ denotes $s(a) = B(a)$ for all $a \in A$ such that $B(a) \in \{T, F\}$.

There are only 3^n 3-valued belief states for n state variables. Because no sequential plan for reaching a belief state has to visit any belief state more than once, such plans have length at most 3^n .

```

procedure 3reachG( $A, I, O, G, B, s, B', s', m$ )
if  $m \leq 0$  then
  if  $(B = B'$  and  $s = s')$  or  $\exists \text{app}(A, I, o, B, s, B', s')$  for some  $o \in O$  and  $\{s, s'\} \cap G \neq \emptyset$  then return true
  else return false
else
  for each  $B'' : A \rightarrow \{T, F, U\}$  and state  $s'' \in B''$  do
    if 3reachG( $A, I, O, G, B, s, B'', s'', m - 1$ ) and 3reach( $A, I, O, G, B'', s'', B', s', m - 1$ ) then return true;
    if 3reach( $A, I, O, G, B, s, B'', s'', m - 1$ ) and 3reachG( $A, I, O, G, B'', s'', B', s', m - 1$ ) then return true
  end for
  return false

```

Fig. 14. Algorithm for finding a path from B to B' that visits G

```

procedure 3reach( $A, I, O, G, B, s, B', s', m$ )
if  $m \leq 0$  then
  if  $(B = B'$  and  $s = s')$  or  $\exists \text{app}(A, I, o, B, s, B', s')$  for some  $o \in O$  then return true
  else return false
else
  for each  $B'' : A \rightarrow \{T, F, U\}$  and  $s'' \in B''$  do
    if 3reach( $A, I, O, G, B, s, B'', s'', m - 1$ ) and 3reach( $A, I, O, G, B'', s'', B', s', m - 1$ ) then return true
  end for;
  return false

```

Fig. 15. Algorithm for finding a path from B, s to B', s'

```

procedure planexistsPOsi( $A, I, O, G, V$ )
if for every  $B : A \rightarrow \{T, F, U\}$  such that  $B(p) \in \{T, F\}$  for all  $p \in V$  and
   $I(p) \in \{T, F\}$  implies  $B(p) = I(p)$  for all  $p \in A$ ;
  for some  $B_0 : A \rightarrow \{T, F, U\}$  and  $s_0 \in B_0$ 
    3reach( $A, I, O, G, B, s, B_0, s_0, m$ ) and
    for every  $s' \in B_0$ 
      for some  $s'' \in B_0$ 
        3reachG( $A, I, O, G, B_0, s', B_0, s'', m$ )
  then return true

```

Fig. 16. Algorithm for testing plan existence for deterministic state-independent operators and partial observability

observability	transition type		
	state-independent deterministic (ID)	state-dependent deterministic (D)	state-dependent non-deterministic (ND)
full (FO)	PSPACE (T18)	PSPACE (T18)	EXP (T22)
no (UO)	PSPACE (T18)	EXPSPACE (T20)	EXPSPACE (T20)
partial (PO)	PSPACE (T18)	EXPSPACE (T20)	2-EXP (T23)

Table I. Complexity lower bound (hardness) theorems for reachability and several initial states (PLANSAT-??-??-RG)

observability	transition type		
	state-independent deterministic (ID)	state-dependent deterministic (D)	state-dependent non-deterministic (ND)
full (FO)	PSPACE (T18)	PSPACE (T18)	EXP (T22)
no (UO)	PSPACE (T18)	PSPACE (T18)	EXPSPACE (T20)
partial (PO)	PSPACE (T18)	PSPACE (T18)	2-EXP (T23)

Table II. Complexity lower bound (hardness) theorems for reachability and one initial state (PLANSAT-??-??-RG-1)

Testing the existence of a loop that visits a goal state by the procedure *3reach* is done for each starting belief state B of a loop and each state $s \in B$ separately. The desired loop is then obtained by juxtaposing the individual loops. The idea of the procedure is that the loop has to be executable for any starting state in B , and when starting in s a goal state will be visited.

Memory consumption at each call is only polynomial in the number of state variables, and the recursion depths are only polynomial. Hence the whole memory consumption is only polynomial in the size of the succinct transition system and the plan existence problem is in PSPACE. \square

7. SUMMARY OF THE RESULTS

We summarize the results of this work and results established in earlier work.

7.1 Lower bounds for reachability

On goal reachability we have hardness proofs for certain complexity classes for three different degrees of observability (unobservable, full, partial) and transitions with three types of restrictions (nondeterministic, deterministic state-dependent, deterministic state-independent). References to theorems are given in Tables I and II, the first for succinct transition systems with several initial states, and the second with one initial state.

7.2 Upper bounds for repeated reachability

For repeated reachability we have constructive proofs of membership in the same complexity classes we already proved hardness for goal reachability. These results show that on both of these objectives, the plan existence problems are complete for the complexity classes in questions. References to theorems are given in Tables III and IV.

Comments on the problem complexities. All the PSPACE cases established by Theorem 24 are easily reducible to s-t-reachability in the succinct transition system, accompanied by an iteration over all possible initial and/or goal states. So in these cases the plan/controller synthesis problem is much easier.

observability	transition type		
	state-independent deterministic (ID)	state-dependent deterministic (D)	state-dependent non-deterministic (ND)
full (FO)	PSPACE (T24)	PSPACE (T24)	EXP (T27)
no (UO)	PSPACE (T32)	EXPSPACE (T28)	EXPSPACE (T28)
partial (PO)	PSPACE (T32)	EXPSPACE (T31)	2-EXP (T29)

Table III. Complexity upper bound theorems for repeated reachability and several initial states (PLANSAT-??-??-RRG)

observability	transition type		
	state-independent deterministic (ID)	state-dependent deterministic (D)	state-dependent non-deterministic (ND)
full (FO)	PSPACE (T24)	PSPACE (T24)	EXP (T27)
no (UO)	PSPACE (T24)	PSPACE (T24)	EXPSPACE (T28)
partial (PO)	PSPACE (T24)	PSPACE (T24)	2-EXP (T29)

Table IV. Complexity upper bound theorems for repeated reachability and one initial state (PLANSAT-??-??-RRG-1)

8. IMPLICATIONS TO DISCRETE EVENT SYSTEMS

Discrete event systems with a form of succinct representation can be embedded in the framework used in this work.

We formalize every (controllable or uncontrollable) transition as a deterministic effect. Let ϕ be a formula that defines a set of states. Let e_1, \dots, e_n be the controllable transitions and $e'_1, \dots, e'_{n'}$ the uncontrollable ones applicable in any state satisfying ϕ .

Let x_1, \dots, x_n be auxiliary state variables. We define the following operators for the transitions applicable in states satisfying ϕ .

$$\begin{aligned}
 \langle \top, x_i \rangle & && \text{for every } i \in \{1, \dots, n\} \\
 \langle \top, \neg x_i \rangle & && \text{for every } i \in \{1, \dots, n\} \\
 \langle \phi, (x_1 e_1 | x_2 e_2 | \dots | x_n e_n | \top e'_1 | \top e'_2 | \dots | \top e'_{n'}) \rangle & &&
 \end{aligned}$$

The operators $\langle \top, x_i \rangle$ and $\langle \top, \neg x_i \rangle$ are used for enabling and disabling transitions before the application of the third operator which represents all the (enabled and disabled) controllable and uncontrollable transitions.

Notice that the operators are deterministic only if there are no uncontrollable transitions and at most one controllable transition for every state, and this controllable transition is deterministic. Alternatively, a plan will be deterministic if there are no uncontrollable transitions and at most one controllable transition, that is also deterministic, is enabled at a time.

The definition of succinct DES covers n -safe Petri nets and VASS because n -valued state variables can be easily represented in terms of Boolean variables.

Theorem 33 *Testing the existence of a controller satisfying the goal reachability or the repeated reachability criterion is 2-EXP-complete.*

PROOF. Operators in proof of Theorem 21 are of the form $\langle \phi, (x_1 e_1 | \dots | x_n e_n | \top e'_1 | \dots | \top e'_{n'}) \rangle$. Specifically, $n = 0$, and for operators representing transitions from existential states there is only one effect e_1 , and for operators representing transitions from universal states there may be several. Hence the 2-EXP-hardness proof directly works also for succinct DES. Membership in 2-EXP is by Theorem 29. \square

9. RELATED WORK

AI planning and the problems discussed in this work are closely related to many other areas. In this section we briefly discuss the closest related computational problems and review the main works on computational complexity that are related to ours.

9.1 Controller synthesis for discrete event systems

Complexity of controller synthesis for discrete event systems has earlier been analyzed for DES that are represented enumeratively, that is, without a representation that uses state variables, Petri nets, or vector addition systems. [Tsitsiklis 1989; Rudie and Willems 1995].

9.2 Planning

The simplest plan synthesis problem is the one with one initial state and deterministic operators. This problem is essentially the s - t -reachability problem of succinctly represented graphs which is PSPACE-complete [Papadimitriou and Yannakakis 1986; Lozano and Balcázar 1990]. A typical succinct representation of graphs uses Boolean circuits for representing the transition matrices. In AI planning the operator-based representation of graphs (transition systems) is typically used, and the problem is PSPACE-complete also for this representation [Bylander 1994]. In AI planning, sets of operators are often represented as operator schemata that are instantiated with some set of constant symbols. This increases the complexity and the plan existence problem for schematic operators without function symbols is EXPSPACE-complete [Erol, Nau, and Subrahmanian 1995]. Allowing function symbols makes the problem undecidable. Erol et al. consider also many other variants of the basic problem.

The non-probabilistic plan existence problem without observability is EXPSPACE-complete [Haslum and Jonsson 2000]. This is the result for which we presented a new proof in Theorem 20. The proof by Haslum and Jonsson is based on a reduction from the universality problem of regular expressions of finite automata with exponentiation [Hopcroft and Ullman 1979]. The EXP-completeness of planning with full observability follows from results of Littman [1997] which we will discuss below.

De Giacomo and Vardi [2000] analyze a number of planning problems in an automata-theoretic framework. These problems are similar to the ones addressed by our work, but the formal frameworks differ. The planning objectives are represented as formulae in a temporal logic, but only deterministic actions are considered. One of De Giacomo and Vardi's results has a counterpart in our work: the EXPSPACE-completeness of plan existence with deterministic actions and partial observability is Theorem 8 in [Giacomo and Vardi 2000] and our Theorem 30.

Considering the high complexity of planning in the general case and the fact that plans can be extremely big, it is justified to look at the complexity of planning when there are strong restrictions on plan size. A practically relevant restriction is to consider plans of small (polynomial) size. For plans without loops testing whether a given plan reaches a given goal state takes only nondeterministic polynomial time in the size of the plan, even when the transition systems have partial observability. When the plan may only have a size that is polynomial in the size of the transition system, testing the existence of acyclic plans is in NP^{NP} , which can be taken advantage of in plan construction [Rintanen 1999].

Also other types of restrictions have been addressed in earlier work, for example having no restrictions on plan size, and restrictions on the lengths of executions of the plans [Baral, Kreinovich, and Trejo 2000; Turner 2002].

Some of the proofs of Mundhenk et al. [2000] on the complexity of bounded-horizon MDPs are also applicable to non-probabilistic planning problems, for example yielding the NEXP-completeness of planning without observability and at most exponential length executions.

Polynomial time planning algorithms are only possible under very strong syntactic or structural restrictions. Works that identify tractable classes of planning include [Bylander 1994; Bäckström and Nebel 1995].

9.3 MDPs and probabilistic planning

Markov decision processes are a formalization of sequential decision making that is very close to the problems in AI planning. Papadimitriou and Tsitsiklis investigate the complexity of the basic decision problems about (fully observable) MDPs, and show that the problem of existence of a policy with a positive reward is P-complete over the main optimality criteria of finite-horizon, discounted and average rewards [Papadimitriou and Tsitsiklis 1987]. They also show that for partially observable MDPs (POMDPs) the same problem is PSPACE-hard, and in PSPACE if the horizon length does not exceed the number of states. The representation of MDPs Papadimitriou and Tsitsiklis has size that is proportional to the number of states, and with succinct representations the complexities increase. For example, the P-completeness result this way exactly corresponds to the EXP-completeness results of non-probabilistic and probabilistic planning with full observability, as discussed in this work and for example by Littman [1997].

For partially observable MDPs the problem of testing the existence of optimal policies under the main infinite-horizon optimality criteria is undecidable [Madani, Hanks, and Condon 2003]. The proof Madani et al. give is based on the close connection between the plan existence problem without observability and the non-emptiness problem of probabilistic finite automata [Paz 1971; Condon and Lipton 1989].

After recognizing that the general POMDP policy construction problem is not solvable, one can look at restricted variants of the general problem. Mundhenk et al. [2000] analyze the construction of finite-horizon policies for succinctly represented POMDPs under different observability restrictions, and restrictions to polynomial and exponentially long horizons. Their succinct representation of POMDPs is based on encoding transition matrices as Boolean circuits. While for non-probabilistic planning in the fully observable, unobservable and in the general case the plan existence problems are respectively complete for EXP, EXPSPACE and 2-EXP, Mundhenk et al. show that the corresponding policy existence problems for history-dependent POMDPs represented as circuits and with exponentially long horizons are complete for EXP, NEXP and EXPSPACE. The latter complexities are this low because of the restriction to exponential horizon length. Planning problems with no observability or partial observability may have only solutions with horizons of doubly exponential length.

With the restriction to plans with success probability 1, as in the problems in the present work, considering a cost measure and finding plans that minimize the expected cost or have a expected cost below a given constant also makes the plan existence problem undecidable.

Theorem 34 *Testing existence of plans with success probability 1 and expected reward exceeding a constant c for probabilistic planning with partial observability and the goal reachability criterion is undecidable.*

PROOF. We sketch a simple reduction from the unsolvable planning problem considered by Madani et al. [2003].

Take a problem instance π_1 of unobservable probabilistic planning characterized by a set O of probabilistic operators, a set I of initial states, and a set G of goal states.

We construct a new problem instance π_2 by adding a new operator o_1 and one new state s_g that is not reachable by any operator in O . The operator o_1 reaches s_g from any non-goal state with cost 1 and from any goal state with cost 0. All operators in O are applicable in s_g and go to some other state from s_g .

The problem of testing whether a plan with success probability greater than p is unsolvable. We reduce this problem to testing whether a plan with success probability 1 and expected cost less than $c = 1 - p$ exists.

Assume there is a plan for π_1 with success probability $> c$. This plan is a sequence of operators. Add o_1 to the end of this sequence. Now the plan solves π_2 with probability 1 and expected cost $c = 1 - p$.

Assume there is a plan for π_2 with success probability 1 and expected cost $c = 1 - p$. This plan is a sequence of operators and o_1 is the last operator. Construct a new plan by deleting o_1 . Now this new plan solves π_1 and has success probability $> p$. \square

9.4 Game theory

Many of the problems in this work have a distinct game theoretic character because non-determinism can be understood as an opponent whose behavior cannot be controlled. This together with the requirement that the goals are reached with probability 1 make the problems considered by us very game-theoretic, and the plan existence problems – in cases when unbounded looping is not allowed – are equivalent to the game-theoretic problem of determining the existence of winning strategies. So, our results directly yield lower and upper bounds for the complexities of these game theoretic problems under different observability restrictions.

Notice that in many practically interesting game-theoretic problems, for example card games like poker, winning strategies do not exist, and the practically interesting computational problem is finding strategies that are optimal in the sense that they produce the highest expected payoff. This problem requires considering probabilities. As implied by the results of Madani et al. [2003], in the kind of general framework considered in this work with games of unbounded length the problem of determining whether there is a strategy with a given expected payoff is undecidable.

The complexity of existence of strategies having a given payoff has earlier been investigated in games represented in the strategic and extensive forms, see for example [Gilboa and Zemel 1989; Koller and Megiddo 1992], as well as existence of winning strategies in simple fully-observable stochastic games [Condon 1992]. Extensive form is a game representation that is much less succinct than the kind of state variable based representations considered in this work. The extensive form encodes all the belief states of the players explicitly, and the number of relevant belief states may be exponential in the number of state variables. [Zwick and Paterson 1996]

Because nondeterminism can be viewed as the freedom of action of another agent, our results on the complexity of the plan existence problems with nondeterministic operators directly yields the complexity of the corresponding problem of existence of winning strategies for 2-player games. Specifically, our Theorem 21 shows that the problem of existence of winning strategies for succinctly represented 2-player games with partial observability is 2-EXP-hard.

9.5 Temporal logic program synthesis

Different types of controller synthesis problems for temporal logic specifications have been considered in earlier work. The planning problems in this work most closely match the open system synthesis problems in which the controller interacts with an external uncontrollable environment. Early works on this topic include Pnueli and Rosner [1989]. It is assumed that the current state of the system is unambiguously known, and therefore the problem corresponds to planning with full observability as considered in this work.

[1995]

Kupferman and Vardi [1997] have considered the open systems controller synthesis problem under partial observability when the synthesis problem is expressed in the branching-time temporal logics CTL and CTL*. The synthesis problems for CTL and CTL* are complete respectively for the complexity classes EXP and 2-EXP. The satisfiability problems for CTL and CTL* are also complete for the same complexity classes [Vardi and Stockmeyer 1985]. That CTL synthesis is EXP-complete means that the 2-EXP-complete partially observable controller synthesis problem cannot be efficiently translated into CTL.

10. CONCLUSIONS

The proofs of many complexity results also directly yield techniques that allow efficient implementation of some of the restricted plan synthesis problems. In the general case, belief states may be arbitrary sets of states, and algorithm implementations are forced to use representations like propositional formulae for representing one belief state. Reasoning with such representations is very expensive. For example, testing the non-emptiness of a belief state represented by a propositional formula is NP-hard. Like shown by our results, for example in the state-independent deterministic case a belief state can instead be represented as a vector of values true, false and unchanged, one vector element for every state variable. This directly yields a considerable computational advantage over more general belief state representations.

Acknowledgments

The research was partly supported by DFG grant RI 1177/2-1.

References

- BÄCKSTRÖM, C. AND NEBEL, B. 1995. Complexity results for SAS⁺ planning. *Computational Intelligence* 11, 4, 625–655.
- BALCÁZAR, J. L., DÍAZ, J., AND GABARRÓ, J. 1988. *Structural Complexity I*. Springer-Verlag, Berlin.
- BALCÁZAR, J. L., DÍAZ, J., AND GABARRÓ, J. 1990. *Structural Complexity II*. Springer-Verlag, Berlin.
- BARAL, C., KREINOVICH, V., AND TREJO, R. 2000. Computational complexity of planning and approximate planning in the presence of incompleteness. *Artificial Intelligence* 122, 1, 241–267.
- BERTOLI, P., CIMATTI, A., ROVERI, M., AND TRAVERSO, P. 2001. Planning in nondeterministic domains under partial observability via symbolic model checking. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, B. Nebel, Ed. (2001). Morgan Kaufmann Publishers, 473–478.

- BONET, B. AND GEFFNER, H. 2000. Planning with incomplete information as heuristic search in belief space. In *Proceedings of the Fifth International Conference on Artificial Intelligence Planning Systems*, S. Chien, S. Kambhampati, and C. A. Knoblock, Eds. (2000). AAAI Press, 52–61.
- BYLANDER, T. 1994. The computational complexity of propositional STRIPS planning. *Artificial Intelligence* 69, 1-2, 165–204.
- CHANDRA, A., KOZEN, D., AND STOCKMEYER, L. 1981. Alternation. *J. ACM* 28, 1, 114–133.
- CHO, H. AND MARCUS, S. 1989. On supremal languages of classes of sublanguages that arise in supervisor synthesis problems with partial observation. *Mathematics of Control, Signals, and Systems* 2, 47–69.
- CIMATTI, A., PISTORE, M., ROVERI, M., AND TRAVERSO, P. 2003. Weak, strong, and strong cyclic planning via symbolic model checking. *Artificial Intelligence* 147, 1–2, 35–84.
- CONDON, A. 1992. The complexity of stochastic games. *Information and Computation* 96, 2, 203–224.
- CONDON, A. AND LIPTON, R. J. 1989. On the complexity of space bounded interactive proofs (extended abstract). In *Proceedings of the 30th IEEE Symposium on Foundations of Computer Science* (1989). IEEE, 462–467.
- DENHAM, M. J. 1988. A Petri-net approach to the control of discrete-event systems. In *Advanced Computing Concepts and Techniques in Control Engineering*, M. J. Denham and A. J. Laub, Eds., *NATO ASI Series* vol. F47, 192–214. Berlin: Springer-Verlag.
- EROL, K., NAU, D. S., AND SUBRAHMANIAN, V. S. 1995. Complexity, decidability and undecidability results for domain-independent planning. *Artificial Intelligence* 76, 1–2, 75–88.
- GHALLAB, M., HOWE, A., KNOBLOCK, C., MCDERMOTT, D., RAM, A., VELOSO, M., WELD, D., AND WILKINS, D. 1998. PDDL - the Planning Domain Definition Language, version 1.2. Technical Report CVC TR-98-003/DCS TR-1165 (Oct.), Yale Center for Computational Vision and Control, Yale University.
- GIACOMO, G. D. AND VARDI, M. Y. 2000. Automata-theoretic approach to planning for temporally extended goals. In *Recent Advances in AI Planning, Fifth European Conference on Planning (ECP'99)*, S. Biundo and M. Fox, Eds., *Lecture Notes in Artificial Intelligence* no. 1809 (2000). Springer-Verlag, 226–238.
- GILBOA, I. AND ZEMEL, E. 1989. Nash and correlated equilibria: Some complexity considerations. *Games and Economic Behavior* 1, 80–93.
- HASLUM, P. AND JONSSON, P. 2000. Some results on the complexity of planning with incomplete information. In *Recent Advances in AI Planning, Fifth European Conference on Planning (ECP'99)*, S. Biundo and M. Fox, Eds., *Lecture Notes in Artificial Intelligence* no. 1809 (2000). Springer-Verlag, 308–318.
- HOPCROFT, J. AND PANSIOT, J.-J. 1979. On the reachability problem for 5-dimensional vector addition systems. *Theoretical Computer Science* 8, 135–159.
- HOPCROFT, J. E. AND ULLMAN, J. D. 1979. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley Publishing Company.
- HOWARD, R. A. 1960. *Dynamic programming and Markov decision processes*. The MIT Press.
- ICHIKAWA, A. AND HIRAIISHI, K. 1988. Analysis and control of discrete-event systems represented as Petri nets. In *Discrete Event Systems: Models and Applications, IIASA Conference, Sopron Hungary, August 3-7, 1987*, P. Varaiya and B. Kurzhanski, Eds., *Lecture Notes in Control and Information Sciences* no. 103 (1988). Springer-Verlag, 115–134.
- INAN, K. 1994. Nondeterministic supervision under partial observations. In *The 11th International Conference on Analysis and Optimization of Systems, Discrete Event Systems*, G. Cohen and J. P. Quadrat, Eds., *Lecture Notes in Computer and Information Sciences* vol. 199 (1994). Springer-Verlag, 39–48.
- KAELBLING, L. P., LITTMAN, M. L., AND CASSANDRA, A. R. 1998. Planning and acting in partially observable stochastic domains. *Artificial Intelligence* 101, 1-2, 99–134.
- KOLLER, D. AND MEGIDDO, N. 1992. The complexity of two-person zero-sum games in extensive form. *Games and Economic Behavior* 4, 528–552.
- KUMAR, R. AND SHAYMAN, M. 1997. Centralized and decentralized supervisory control of nondeterministic systems under partial observation. *SIAM Journal on Control and Optimization* 35, 363–383.
- KUPFERMAN, O. AND VARDI, M. 1997. Synthesis with incomplete information.
- LI, Y. AND WONHAM, W. M. 1993. Control of vector discrete-event system I - the base model. *IEEE Transactions on Automatic Control* 38, 8, 1214–1227.
- LIN, F. AND WONHAM, W. M. 1988. On observability of discrete-event systems. *Information Sciences* 44, 3, 173–198.

- LITTMAN, M. L. 1997. Probabilistic propositional planning: Representations and complexity. In *Proceedings of the 14th National Conference on Artificial Intelligence (AAAI-97) and 9th Innovative Applications of Artificial Intelligence Conference (IAAI-97)* (Menlo Park, July 1997). AAAI Press, 748–754.
- LOZANO, A. AND BALCÁZAR, J. L. 1990. The complexity of graph problems for succinctly represented graphs. In *Graph-Theoretic Concepts in Computer Science, 15th International Workshop, WG'89*, M. Nagl, Ed., Lecture Notes in Computer Science no. 411 (Castle Rolduc, The Netherlands, 1990). Springer-Verlag, 277–286.
- MADANI, O., HANKS, S., AND CONDON, A. 2003. On the undecidability of probabilistic planning and related stochastic optimization problems. *Artificial Intelligence* 147, 1–2, 5–34.
- MARCHAND, H., BOIVINEAU, O., AND LAFORTUNE, S. 2001. Optimal control of discrete event systems under partial observation. In *Proceedings of the 40th IEEE Conference on Decision and Control, CDC'01* (2001).
- MUNDHENK, M., GOLDSMITH, J., LUSENA, C., AND ALLENDER, E. 2000. Complexity of finite-horizon Markov decision process problems. *J. ACM* 47, 4, 681–720.
- MURATA, T. 1989. Petri nets: properties, analysis and applications. *Proceedings of IEEE* 77, 3, 541–580.
- PAPADIMITRIOU, C. H. 1994. *Computational Complexity*. Addison-Wesley Publishing Company.
- PAPADIMITRIOU, C. H. AND TSITSIKLIS, J. N. 1987. The complexity of Markov decision processes. *Mathematics of Operations Research* 12, 3 (Aug.), 441–450.
- PAPADIMITRIOU, C. H. AND YANNAKAKIS, M. 1986. A note on succinct representations of graphs. *Information and Control* 71, 181–185.
- PAZ, A. 1971. *Introduction to Probabilistic Automata*. Academic Press.
- PETRI, C. A. 1962. *Kommunikation mit Automaten*. Ph. D. thesis, Institut für instrumentelle Mathematik, Bonn, Germany.
- PNUELI, A. AND ROSNER, R. 1989. On the synthesis of an asynchronous reactive module. In *Automata, Languages and Programming, 16th International Colloquium*, G. Ausiello, M. Dezani-Ciancaglini, and S. R. D. Rocca, Eds., *Lecture Notes in Computer Science* vol. 372 (July 1989). Springer-Verlag, 652–671.
- PUTERMAN, M. L. 1994. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons.
- RAMADGE, P. AND WONHAM, W. 1987a. Modular feedback logic for discrete event systems. *SIAM Journal of Control and Optimization* 25, 5 (Sept.), 1202–1218.
- RAMADGE, P. AND WONHAM, W. 1987b. Supervisory control of a class of discrete-event processes. *SIAM Journal of Control and Optimization* 25, 1 (Jan.), 206–230.
- REISIG, W. 1985. *Petri Nets: An Introduction*. EATCS Monographs on Theoretical Computer Science vol. 3. Springer-Verlag.
- RINTANEN, J. 1999. Constructing conditional plans by a theorem-prover. *Journal of Artificial Intelligence Research* 10, 323–352.
- RINTANEN, J. 2002. Backward plan construction under partial observability. In *Proceedings of the Sixth International Conference on Artificial Intelligence Planning Systems*, M. Ghallab, J. Hertzberg, and P. Traverso, Eds. (2002). AAAI Press, 173–182.
- RUDIE, K. AND WILLEMS, J. 1995. The computational complexity of decentralized discrete-event control problems. *IEEE Transactions on Automatic Control* 40, 7 (July), 1313–1319.
- SMALLWOOD, R. D. AND SONDIK, E. J. 1973. The optimal control of partially observable Markov processes over a finite horizon. *Operations Research* 21, 1071–1088.
- SREENIVAS, R. S. AND KROGH, B. 1991. Petri net based models for condition/event systems. In *Proceedings of 1991 American Control Conference*, vol. 3 (Boston, Massachusetts, 1991). 2899–2904.
- STOCKMEYER, L. J. AND CHANDRA, A. K. 1979. Provably difficult combinatorial games. *SIAM Journal on Computing* 8, 2, 151–174.
- TSITSIKLIS, J. N. 1989. On the control of discrete-event dynamical systems. *Mathematics of Control, Signals, and Systems* 2, 95–107.
- TURNER, H. 2002. Polynomial-length planning spans the polynomial hierarchy. In *Logics in Artificial Intelligence, European Conference, JELIA 2002*, Lecture Notes in Computer Science no. 2424 (2002). Springer-Verlag, 111–124.
- VARDI, M. AND STOCKMEYER, L. 1985. Improved upper and lower bounds for modal logics of programs. In *Proceedings of the 17th Annual ACM Symposium on Theory of Computing* (1985). Association for Computing Machinery, 240–251.

- VARDI, M. Y. 1995. An automata-theoretic approach to fair realizability and synthesis. In *Computer Aided Verification, Proceedings of the 7th International Conference*, P. Wolper, Ed., *Lecture Notes in Computer Science* vol. 939 (1995). Springer-Verlag, 267–278.
- WELD, D. S., ANDERSON, C. R., AND SMITH, D. E. 1998. Extending Graphplan to handle uncertainty and sensing actions. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98) and the Tenth Conference on Innovative Applications of Artificial Intelligence (IAAI-98)* (1998). AAAI Press, 897–904.
- ZWICK, U. AND PATERSON, M. S. 1996. The complexity of mean payoff games on graphs. *Theoretical Computer Science* 158, 1–2, 343–359.