Artificial Intelligence ••• (••••) •••–•••

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

Conformant plans and beyond: Principles and complexity

Blai Bonet

Departamento de Computación, Universidad Simón Bolívar, Caracas 89000, Venezuela

ARTICLE INFO

Article history:
Received 16 September 2008
Received in revised form 29 October 2009
Accepted 29 October 2009
Available online xxxx

Keywords:
Planning
Complexity of planning
Partially-observable planning
Non-deterministic planning
Modal logic

ABSTRACT

Conformant planning is used to refer to planning for unobservable problems whose solutions, like classical planning, are linear sequences of operators called linear plans. The term 'conformant' is automatically associated with both the unobservable planning model and with linear plans, mainly because the only possible solutions for unobservable problems are linear plans. In this paper we show that linear plans are not only meaningful for unobservable problems but also for partially-observable problems. In such case, the execution of a linear plan generates observations from the environment which must be collected by the agent during the execution of the plan and used at the end in order to determine whether the goal had been achieved or not; this is the typical case in problems of diagnosis in which all the actions are knowledge-gathering actions.

Thus, there are substantial differences about linear plans for the case of unobservable or fully-observable problems, and for the case of partially-observable problems: while linear plans for the former model must conform with properties in state space, linear plans for partially-observable problems must conform with properties in belief space. This differences surface when the problems are allowed to express epistemic goals and conditions using modal logic, and place the plan-existence decision problem in different complexity classes.

Linear plans is one extreme point in a discrete spectrum of solution forms for planning problems. The other extreme point is contingent plans in which there is a branch point for every possible observation at each time step, and thus the number of branch points is not bounded a priori. In the middle of the spectrum, there are plans with a bounded number of branch points. Thus, linear plans are plans with zero branch points and contingent plans are plans with unbounded number of branch points.

In this work, we lay down foundations and principles for the general treatment of linear plans and plans of bounded branching, and provide exact complexity results for novel decision problems. We also show that linear plans for partially-observable problems are not only of theoretical interest since some challenging real-life problems can be dealt with them.

1. Introduction

Consider the game of Mastermind which is a two-person code-breaking game played by the codemaker and the codebreaker. The game begins when the codemaker chooses a secret code, made of 4 pegs colored from 6 available colors (repetitions allowed), and the task of the codebreaker is to discover the code by questioning the codemaker and assessing his answers. Each question, called a guess, is also a sequence of 4 colored pegs that is answered by the codemaker with two tokens of information: first the number of exact matches in the guess, i.e. the number of pegs of the right color and

E-mail address: bonet@ldc.usb.ve.

0004-3702/$ – see front matter
doi:10.1016/j.artint.2009.11.001

Please cite this article in press as: B. Bonet, Conformant plans and beyond: Principles and complexity, Artificial Intelligence (2009), doi:10.1016/j.artint.2009.11.001

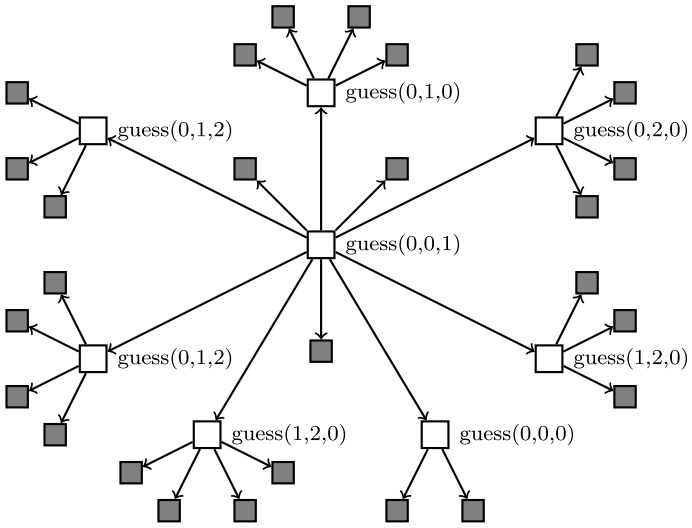

Fig. 1. Optimal contingent plan for Mastermind with 3 pegs and 3 colors.

in the right position, and second the number of near matches in the guess, i.e. the number of pegs of the right color but in wrong position. The codebreaker wins if he can discover the secret code in at most 10 guesses, otherwise the codemaker wins. This popular game has captivated the attention of millions of people [60] including some renowned mathematicians [14,18,37,38].

The game proceeds in guess-and-answer stages in which a guess may depend on the information acquired during the previous stages. A winning strategy for Mastermind can be depicted as a tree whose nodes are the subsets of the possible secret codes at a given stage. For example, the root node is the set of $6^4 = 1296$ possible secret codes since, at the beginning, the codebreaker has no information whatsoever. Each internal node of the tree is labeled with the guess to be done in case the game reaches that stage, and for each possible answer to the guess, there is a child node that corresponds to the subset of secret codes compatible with the answer. The leaves of the tree are nodes that correspond to singletons since these represent stages of the game at which the codebreaker has discovered the secret code.

Fig. 1, for example, depicts an optimal strategy for a game of Mastermind with 3 pegs and 3 colors. Although the labels on the edges that tell the possible answers to the guesses are not shown, the important thing to note is the form of the solution, i.e. its tree structure.

The game of Mastermind can be cast as a non-deterministic planning problem with partial observability, and hence solutions can be obtained with planners that perform search in belief space. Indeed, subsets of possible states (secret codes in Mastermind) are called *belief states*, and solutions like Fig. 1 are called *contingent plans* in belief space or contingent plans with partial observability. In general, if b is a belief state in the solution graph, o is an operator applicable at b, and $z_1, \ldots, z_n$ are the possible observations obtainable after the application of the operator o at b, then the children of b in the solution graph are the beliefs $b_o^{z_1}, \ldots, b_o^{z_n}$ where $b_o^{z_i}$ denotes the belief that results after applying o at b and observing z_i.

There are natural variations of the game such as the ones that result by increasing the number of colors in the secret code or the number of available colors to choose from. However, there is a more interesting variation that is known as *static Mastermind* [14,23]. In this variation, the codebreaker is asked to give ahead a linear sequence of guesses such that the secret code can be determined from the answers upon such guesses. For example, for a game with 3 pegs and 3 colors, the sequence

$$\sigma = \langle \text{guess}(2,0,0), \text{guess}(2,1,0), \text{guess}(2,2,1) \rangle$$

is guaranteed to succeed independently of the chosen secret code. Moreover, its length is minimum among all such sequences, and hence corresponds to an optimal sequence. Also observe that the length of σ is greater than or equal to the length of any branch in the optimal contingent plan in Fig. 1; this is not a coincidence since the sequence σ must discover the secret code independently of its value.

We call a sequence like σ a linear or conformant plan for a partially-observable problem. Linear plans had been studied before in the context of unobservable planning by the name of conformant planning [16,21,24,30,57]. Unlike partially-observable problems like Mastermind, unobservable problems only admit linear plans as solutions, since under the hypothesis of null observability, the decision maker has no available input on which to base his decisions. In unobservable problems, a linear plan generates a collection of trajectories that result from the uncertainty in the initial state and the non-determinism in the actions of the problem. A linear plan is a valid plan when each trajectory that starts at an initial state ends in a goal state. This is the reason for the name 'conformant' as it means that the trajectories generated by the plan conform with the goal of the problem.

Please cite this article in press as: B. Bonet, Conformant plans and beyond: Principles and complexity, Artificial Intelligence (2009), doi:10.1016/j.artint.2009.11.001

Until now, linear plans had been only generally considered for unobservable planning problems. However, as the example of Mastermind shows, it also makes sense to consider linear plans for problems with partial observability (and also for problems with full observability). Hence, we think that the term 'conformant' has been improperly used to refer to two different things: a class of solutions, namely linear plans, and to the class of unobservable planning problems.

In fully-observable domains, a linear plan does not make use of the information available at each time step as the plan determines the actions to do at each step independently of contingencies. Thus, this case is essentially the same as the unobservable one and, from now on, we treat unobservable problems as non-deterministic fully-observable problems for which only linear plans are acceptable. In partially-observable domains, a linear plan also dictates the actions to do at each time step, without deviating from a unique line of execution, yet there is an important difference with respect to the former case as now the decision maker must collect the observations generated during the application of the plan in order to achieve the goal at the end of the plan.

There are substantial differences between the cases of linear plans for fully-observable and partially-observable domains. The first important difference is about the guarantees offered by such a plan. In the former case, the agent has the guarantee that the last state after the application of the plan will be a goal state. In the latter case however, as illustrated in the example of Mastermind, the agent has the guarantee that after applying the plan and collecting the observations generated along, he will have enough information to achieve the goal, e.g., computing the secret code in Mastermind. Hence, *while conformance in fully-observable domains is about the compliance of a set trajectories in state space, conformance for partially-observable domains is about the compliance of a set of trajectories in belief space.*

The second important difference appears when studying the computational complexity of decision problems for both notions of conformance: checking the existence of a linear plan for a fully-observable or unobservable problem is known to be EXPSPACE-complete [27], while checking the existence of a linear plan for a partially-observable problem will be shown to be 2EXPSPACE-complete. This is an important difference since these two complexity classes are known to be different and hence a 2EXPSPACE-complete problem cannot be reduced in polynomial time to an EXPSPACE-complete problem.

It seems that linear plans have not been considered so far for partially-observable problems because the standard representation languages are not able to express epistemic goals and conditions. Indeed, a problem like Mastermind cannot be expressed in such languages as they do not allow to express goals such as 'to know the color of the pegs'. Thus, in order to do a proper study of linear plans, we extend the standard propositional language used in planning with a simple modal operator that can express such goals.

Linear plans is one extreme point in a discrete spectrum of solution forms. The other extreme point is contingent plans in which there is a branch point for every possible observation received at each time step, and thus the number of branch point is not bounded a priori. In the middle of the spectrum, there are plans with a bounded number of branch points; e.g., plans with at most 1 branch point, plans with at most 2 branch points, and so on. Linear plans are plans with no branch points whereas contingent plans are plans with unbounded number of branch points.

The idea of having plans with a bounded number of branches is not new. Meuleau and Smith [45] considered plans of bounded branching. Their work seems to be the first explicit and general treatment of the subject in the area of automated planning. Although their formulation is correct for the fully-observable case, it misses important aspects of the partially-observable case.

Linear plans and plans of bounded branching are also of practical interest. Meuleau and Smith list three practical reasons of why such plans are of interest for NASA: (1) plans must be simple enough so that humans can easily display, understand and be able to modify them, (2) plans may be subject to detailed analysis of resource consumptions, dissipation of energy, etc. and thus be simple for performing such analyses, and (3) rovers and voyagers have limited communication bandwidth, computational resources and storage on board so simple plans are preferred to more complex ones. In areas related to medicine, for example, there are the so-called "diagnostic-test sequences" which can be thought as linear plans for a partially-observable domain. A diagnostic-test sequence is a predetermined set of tests that are performed during a given procedure or on a given sample even if the results of some of the tests end up not being used. For example, when performing blood tests, medical doctors order the so-called blood profiles that consist of a suite of tests to be performed on a single sample of blood. Diagnostic-test sequences exist for two main reasons: a blood sample must be processed within a time frame and thus tests cannot wait for the results of previous tests, and it is not ethically accepted to disturb a patient multiple times by taking one blood sample per test. The situation becomes more dramatic as more invasive procedures are required. The design of diagnostic-test sequences is a relevant area in medicine.

Problems involving rovers also provide real-life and vivid examples of situations that are meaningful to model as partially-observable problems on which to compute linear plans. Indeed, consider a rover in a remote location with the goal of performing scientific tests to determine the presence of a chemical compound. Further, assume that the vehicle has limited capabilities for performing tests, and for analyzing and communicating the results of the tests. In particular, it may be the case that the robot does not have enough computational resources to update its belief state with the results of the tests, nor to send the observations to the base in order to receive new instructions at each time step. In this case, one is interested in computing a linear plan with the property that upon the results from the tests, collected and transmitted back to the base, a team of experts, provided with enough time and resources, would be able to detect the presence or absence of the compound. Thus, knowing from advance that the robot is not able to properly maintain a belief state, there is no

Please cite this article in press as: B. Bonet, Conformant plans and beyond: Principles and complexity, Artificial Intelligence (2009), doi:10.1016/j.artint.2009.11.001

Table 1
Complexity results for fully-observable planning problems. Each cell contains the class for which the corresponding problem is complete and a reference for the result. The last two problems are only defined for problems with no modal formulae; this is indicated with the subscript '*PL*'.

Problem	Without modal formulae	With modal formulae
PLAN-FO-CONT	EXP [53]	EXP (new)
PLAN-FO-LINEAR	EXPSPACE [27]	EXPSPACE (new)
PLAN-FO-BRANCH(k)	EXPSPACE (new)	EXPSPACE (new)
PLAN-FO-BRANCH	EXPSPACE (new)	EXPSPACE (new)
PLAN-FO-BRANCH-LEN$_{PL}$(k)	Σ^P_{2k+2} (new)	n/a
PLAN-FO-BRANCH-LEN$_{PL}$	PSPACE (new)	n/a

need to model the computational limitations of the robot. Instead, one can just focus on modeling the essential parts of the task and on obtaining a linear plan.

In this paper we study linear plans and plans of bounded branching for fully-observable and partially-observable planning problems. We extend the standard propositional language for planning with a modal operator that permits the specification of epistemic goals and preconditions, and perform a thorough complexity analysis on novel decision problems related to the existence of different forms of solutions.

The paper is organized as follows. First, we provide a summary of the complexity results in the following subsection for the convenience of the reader. Then, the propositional modal logic, the planning language and definitions of plans, some examples, and the decision problems are presented in Section 2 through Section 5. These sections are written for a general audience that has some knowledge on representation languages and automated planning. The following sections however, from Section 6 through Section 10, are more technical as they contain the proofs of the complexity results: Section 6 contains inclusion results, Section 7 describes regular expressions with exponentiation and non-deterministic finite automata with counters that are the main tool for showing hardness results, Section 8 shows how counters of double-exponential capacity can be encoded using belief states, and Sections 9 and 10 contain the hardness results and special cases for plans of polynomial length for problems with no modal formulae. Section 11 concludes with a brief discussion.

This paper contains and extends results from the article appeared in the proceedings of the 16th International Conf. on Automated Planning and Scheduling 2006 [5]. This revised extension includes improved proofs, new decision problems, and fixes some errors.

1.1. Summary of complexity results

We study the computational complexity of a number of decision problems for planning with full and partial observability, with and without modal formulae, and with restrictions on the number of branch points and the length of the solutions. The decision problems also depend on the type of the solution: either contingent, linear or a solution with a bounded number of branch points. For example, the decision problem PLAN-FO-CONT deals with the existence of contingent plans for fully-observable planning problems, while PLAN-PO-BRANCH(k) deals with the existence of plans with at most k branch points for partially-observable planning problems. Some of the decision problems are standard, others are extensions of standard ones by considering modal formulae, and others are novel.

Table 1 depicts exact complexity results of decision problems for fully-observable problems. Each row in the table corresponds to a type of decision problem and the columns indicate whether the problem is allowed to have modal formulae or not. For example, the cell in the first row and column corresponds to the problem of checking the existence of a contingent plan for fully-observable problems with no modal formulae. Rintanen [53] showed that this problem is EXP-complete. The second column of the first row contains the same decision problem but for problems that may have modal formulae; in this case, the decision problem is new but its complexity is the same. In this table, CONT refers to contingent, BRANCH(k) refers to at most k branch points, BRANCH refers to problems whose instances are pairs $\langle P, k\rangle$ where P is a planning problem and k is an integer written in binary that bounds the number of branch points, and the subscript *PL* refers to a class of problems that have no modal formulae (defined in Section 10). Precise definitions for the decision problems are given in Sections 5 and 10. Complexity results for partially-observable problems are shown in Table 2.

As we can see, there are novel decision problems and interesting complexity results. Among them, the completeness result for 2EXPSPACE (in Table 2) shows that computing linear plans for partially-observable problems is quite challenging. It is worth noticing that although this result is new, it is not unexpected given the known results for fully-observable problems. Indeed, for fully-observable problems, we know that checking the existence of a contingent plan is EXP-complete while EXPSPACE-complete for a linear plan. Thus, since checking the existence of a contingent plan for partially-observable problems is 2EXP-complete [53], it is not surprising that the complexity rises to 2EXPSPACE when checking the existence of a linear plan.

Please cite this article in press as: B. Bonet, Conformant plans and beyond: Principles and complexity, Artificial Intelligence (2009), doi:10.1016/j.artint.2009.11.001

Table 2
Complexity results for partially-observable planning problems. Each cell contains the class for which the corresponding problem is complete and a reference for the result. The last two problems are only defined for problems with no modal formulae; this is indicated with the subscript '*PL*'.

Problem	Without modal formulae	With modal formulae
PLAN-PO-CONT	2EXP [53]	2EXP (new)
PLAN-PO-LINEAR	EXPSPACE (new)	2EXPSPACE (new)
PLAN-PO-BRANCH(k)	EXPSPACE (new)	2EXPSPACE (new)
PLAN-PO-BRANCH	EXPSPACE (new)	2EXPSPACE (new)
PLAN-PO-BRANCH-LEN$_{PL}$(k)	Σ^P_{2k+2} (new)	n/a
PLAN-PO-BRANCH-LEN$_{PL}$	PSPACE (new)	n/a

2. Propositional modal logic

We use a simple propositional modal language that extends propositional logic with a unary modal operator $\Box$ ('box'). Well-formed formulae (wffs) are build up recursively from propositions and the propositional constant falsum ('$\bot$') using the following rules:

- a proposition p is a wff,
- $\bot$ is a wff,
- if φ is a wff, then $\neg\varphi$ is a wff,
- if φ and ψ are wffs, then $\varphi \vee \psi$ is a wff, and
- if φ is a wff, then $\Box\varphi$ is a wff.

Conjunctions, implications, bi-implications and the constant true ('$\top$') refer to the standard abbreviations: $\varphi \wedge \psi := \neg(\neg\varphi \vee \neg\psi)$, $\varphi \rightarrow \psi := \neg\varphi \vee \psi$, $\varphi \leftrightarrow \psi := (\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)$, and $\top := \neg\bot$. We also consider the dual operator $\Diamond$ ('diamond') of the operator $\Box$ defined by the abbreviation $\Diamond\varphi := \neg\Box\neg\varphi$.

States are interpretations of the propositional symbols. We interchangeably denote states as interpretations and as the subsets of propositions that they make true. Belief states refer to collection of states and are often represented as subsets of subsets of propositions. As it is standard in modal logic [4,34] and theories for reasoning about knowledge [19], the semantics of modal formulae is given in terms of Kripke models and frames. In our case, we are only interested in interpreting formulae with respect to belief states and thus provide the semantics in terms of pairs (b, s), where b is a belief state and s is a state, using structural induction as follows:

- $(b, s) \models p$ iff $p \in s$,
- $(b, s) \models \bot$ never,
- $(b, s) \models \neg\varphi$ iff not $(b, s) \models \varphi$,
- $(b, s) \models \varphi \vee \psi$ iff $(b, s) \models \varphi$ or $(b, s) \models \psi$, and
- $(b, s) \models \Box\varphi$ iff $(b, s') \models \varphi$ for all $s' \in b$.

A formula φ is *globally true* in belief state b, written as $b \models \varphi$, if it is satisfied at all states; i.e. $(b, s) \models \varphi$ for all $s \in b$. This notion of interpretation is a specialization of the standard Kripke semantics for modal logic in which the set of nodes (in the frame) is the belief state b, and the accessibility relation (in the frame) is a universal relation over nodes; i.e., every node is related to each other. A frame of this type is in turn a special case of frames with an accessibility relation that is an equivalence relation. A sound and complete axiomatization for this class of frames is the system **S5**, yet we will be only concerned with being able to interpret formulae at the model-theoretic level with respect to belief states. The reader interested in a thorough treatment of modal logic is referred to the work of Blackburn, de Rijke, and Venema [4].

From a computational perspective, we are interested in how to decide whether $b \models \varphi$ for a given belief state b and wff φ, and what is the cost of doing it. It turns out that for our purposes it suffices to know that such decision can be computed in polynomial space in the size of the belief state and φ. Indeed, a simple recursive algorithm that tests '$(b, s) \models \varphi$' by decomposing φ into sub-formulae can be used to test $b \models \varphi$. Such an algorithm only needs to store a stack of depth $O(|\varphi|)$.

Modal logic is more expressive than propositional logic as it permits, for example, to express preconditions for actions or goals of the form 'know φ', which holds when all states in the belief state agree on the interpretation of φ, and 'possibly φ' which holds when some state in the belief state satisfies φ. Indeed, 'know φ' can be defined as an abbreviation of $\Box\varphi \vee \Box\neg\varphi$ while 'possibly φ' can be defined as an abbreviation of $\Diamond\varphi$.

Example: Goal for the game of Mastermind

Later we provide a complete formulation of the game of Mastermind in which the secret code is represented using propositions $s_{i,j}$ that are true in a state s when the ith peg is of color j. In Mastermind the goal is to reach a belief state in which the interpretations of all propositions denoting the secret code are known to the codebreaker. Hence, the goal is defined with the formula $\bigwedge_{1 \leqslant i,j \leqslant 3}(\Box s_{i,j} \vee \Box\neg s_{i,j})$ for a game of Mastermind with 3 pegs and 3 colors.

Please cite this article in press as: B. Bonet, Conformant plans and beyond: Principles and complexity, Artificial Intelligence (2009), doi:10.1016/j.artint.2009.11.001

3. Planning problems, belief states and plans

We consider planning problems defined by tuples of the form $P = \langle D, I, G, O, Z\rangle$ where D is a set of propositional symbols (fluents), I is a propositional formula without modalities that defines the initial states, G is a propositional formula that defines the goal states, O is a set of operators, and $Z \subseteq D$ is the subset of *observable* fluents. Operators are pairs $\langle \varphi, \alpha\rangle$ where φ, called the precondition, is a propositional formula and α is an effect defined as follows:

- $\top$ is the null or empty effect,
- p and $\neg p$ are atomic effects for $p \in D$,
- if $\alpha_1, \ldots, \alpha_n$ are effects, then $(\alpha_1 \wedge \cdots \wedge \alpha_n)$ is a parallel effect,
- if $\alpha_1, \ldots, \alpha_n$ are effects, then $(\alpha_1 \oplus \cdots \oplus \alpha_n)$ is a non-deterministic effect, and
- if φ is a formula and α is an effect, then $(\varphi \rhd \alpha)$ is a conditional effect where φ and α are called the condition and effect of the conditional effect.

Both preconditions and conditions may contain modal operators. However, there is an important difference in how they are evaluated since the former are evaluated at belief states while the latter at states within a belief state. This is the standard semantics in planning for conditional operators as it makes possible for an effect to trigger at some states and not at others in the same belief state.

As said before, states for the planning problem are valuations for the propositions in D and belief states are subsets of states; their maximum number is $2^{|D|}$ for states and $2^{2^{|D|}}$ for beliefs states. An operator $o = \langle \varphi, \alpha\rangle$ is applicable at the belief state b iff $b \models \varphi$; in such case the states that result from the application of α at each state $s \in b$ is the belief state

$$Res(o, b) \stackrel{\text{def}}{=} \bigcup_{s \in b} Appl\big(Eff(\alpha|b, s), s\big),$$

where $Eff(\alpha|b, s)$ is the set of *atomic effects* induced by α given (b, s), and $Appl(E, s)$ is the set of states that result from the application of the atomic effects E on state s. If the operator is not applicable the result is undefined. The set $Eff(\alpha|b, s)$ of atomic effects is defined as

- $Eff(\top|b, s) \stackrel{\text{def}}{=} \{\emptyset\}$,
- $Eff(\ell|b, s) \stackrel{\text{def}}{=} \{\{\ell\}\}$ for literal ℓ,
- $Eff(\alpha_1 \wedge \cdots \wedge \alpha_n|b, s) \stackrel{\text{def}}{=} \{\bigcup_{i=1}^{n} E_i\colon\ E_i \in Eff(\alpha_i|b, s)\}$,
- $Eff(\alpha_1 \oplus \cdots \oplus \alpha_n|b, s) \stackrel{\text{def}}{=} \bigcup_{i=1}^{n} Eff(\alpha_i|b, s)$, and
- $Eff(\varphi \rhd \alpha|b, s) \stackrel{\text{def}}{=} Eff(\alpha|b, s)$ if $(b, s) \models \varphi$, and $Eff(\varphi \rhd \alpha|b, s) \stackrel{\text{def}}{=} \{\emptyset\}$ otherwise.

Observe that the effects are calculated with respect to a belief and state. In particular, the condition φ of the effect '$\varphi \rhd \alpha$' is evaluated with respect to the pair (b, s) instead of the belief b. Thus, it is possible that a conditional effect triggers at some state but not at other in the same belief. Finally, the application of a set E of atomic effects on a state s is defined as

$$Appl(E, s) \stackrel{\text{def}}{=} \left\{ e \cup s'\colon\ e \in E,\ s' = s \setminus \bigcup_{p \in \text{prop}(e)} \{p, \neg p\} \right\},$$

where $\text{prop}(e)$ is the set of propositions mentioned in the effect e.

For example, consider the fluents $\{\min, m_1, m_0\}$, the formula $\varphi = \min \wedge m_0 \wedge \Diamond(\min \wedge \neg m_0)$, the effect $\alpha = \varphi \rhd \neg \min$, and the belief $b = \{s_1 = \{m_1\}, s_2 = \{\min, m_0\}, s_3 = \{\min\}\}$. Then,

- $(b, s_1) \not\models \varphi$ since $\min \notin s_1$, hence $Eff(\alpha|b, s_1) = \{\emptyset\}$;
- $(b, s_2) \models \varphi$ since $\{\min, m_0\} \subseteq s_2$ and $(b, s_3) \models \min \wedge \neg m_0$, hence $Eff(\alpha|b, s_2) = \{\{\neg \min\}\}$;
- $(b, s_3) \not\models \varphi$ since $m_0 \notin s_3$, hence $Eff(\alpha|b, s_3) = \{\emptyset\}$.

Thus, $Res(\langle \top, \alpha\rangle, b) = Appl(\{\emptyset\}, s_1) \cup Appl(\{\{\neg \min\}\}, s_2) \cup Appl(\{\emptyset\}, s_3) = \{s_1, \{m_0\}, s_3\}$.

Different classes of models are obtained by controlling the parameters in a problem $P = \langle D, I, G, O, Z\rangle$. If there are no non-deterministic effects, and I determines a unique state, P is an ADL problem [50], or a STRIPS problem [20] if there are no conditional effects and G and all preconditions are conjunctions of atoms. If the set of observables equals D then P is a fully-observable non-deterministic planning problem, which can be thought of as a non-deterministic Markov Decision Process (MDP) [3,52]; in this case, we drop Z from the notation and write $P = \langle D, I, G, O\rangle$. If $Z \neq D$ then P is a contingent planning problem, which can be thought of as a non-deterministic Partially Observable MDP [6,35].

The form of valid plans varies with the nature of the problem. For ADL and unobservable problems, a plan is a linear sequence of operators that achieves the goal no matter what is the initial state and the non-determinism involved, and hence plans can be recovered by search in state space for ADL problems [7,28,31] and search in belief space for linear plans for

Please cite this article in press as: B. Bonet, Conformant plans and beyond: Principles and complexity, Artificial Intelligence (2009), doi:10.1016/j.artint.2009.11.001

unobservable problems [6,10,30,47,54]. Contingent plans are functions that map states into operators for MDPs, and belief states into operators for POMDPs. In these cases, a valid plan can be obtained by different means: dynamic programming over state space or belief space [3,12,52,58], AND/OR search in state or belief space [8,9,26], and other techniques as well [29,33,43]. For fully-observable and partially-observable problems, we also consider linear plans and plans with a bounded number of branch points as defined next.

3.1. Plans

A goal belief state is a belief state that satisfies the goal formula, i.e. $b \vDash G$. If o is an applicable operator at b, we let b_o denote the belief state $Res(o, b)$ that results from the application of o at b. The function $Res(o, b)$ is extended over sequences of operators by $Res(\langle\rangle, b) \stackrel{\text{def}}{=} b$ and $Res(\langle o_1, \ldots, o_n\rangle, b) \stackrel{\text{def}}{=} Res(o_n, Res(\langle o_1, \ldots, o_{n-1}\rangle, b))$, with the proviso that each operator in the sequence is applicable at the corresponding belief, or else the result is undefined.

Let $P = \langle D, I, G, O\rangle$ be a fully-observable problem and b_I the initial belief state; i.e., $b_I \stackrel{\text{def}}{=} \{s\colon\ s \vDash I\}$. A sequence $\pi = \langle o_1, \ldots, o_n\rangle$ of operators is said to be a *linear or conformant* plan for P iff $Res(\pi, b_I)$ is a goal belief.

A plan with bounded branching for P is a tree that is defined with respect to the initial belief. We define plans of bounded branching as follows. A 0-plan for belief b is an applicable sequence of operators that maps b into a goal belief; e.g., a 0-plan for b_I is a linear plan for b_I. In general, a k-plan for b is a labeled and directed tree $T = \langle V, E, r, \ell\rangle$ where $r \in V$ is the root node and ℓ is a labeling function that maps nodes n into operator sequences $\ell(n)$ and edges e into states $\ell(e)$ such that:

- the height of T is less than or equal to k,
- for each node n, the function $\ell(n, \cdot)$ is one-to-one, i.e. $\ell(n, n') = \ell(n, n'')$ implies $n' = n''$,
- the sequence of operators $\ell(r)$ is applicable at b,
- the subset of states $\{\ell(r, n)\colon\ (r, n) \in E\}$ is equal to $Res(\ell(r), b)$, and
- for each edge (r, n), the subtree rooted at n is a $(k-1)$-plan for the singleton $\{\ell(r, n)\}$.

In words, the label of the root denotes the sequence of operators to apply at the initial belief b which results in the subset of states $Res(\ell(r), b)$. Then, the agent 'observes' the world and determines that the current state is some s in $Res(\ell(r), b)$. By the fourth condition, there is a child n of r such that $\ell(r, n) = s$, and thus the subtree rooted at n is a $(k-1)$-plan for the singleton belief state $\{s\}$. In particular, when n is a leaf of the tree, $\ell(n)$ is a linear plan for the singleton $\{s\}$.

These conditions characterize valid plans with at most k branch points for fully-observable problems: a plan with at most k branch points is a k-plan for b_I. A plan with no bounds on branching is just a tree of arbitrary height, and a contingent plan is a plan with no bounds on branching in which each sequence $\ell(n)$ of operators contains only one operator.

At first sight, it may appear that constraints on the number of branch points translate into constraints on the amount of information that an agent needs to store in order to execute the plan. This however is not true because our plans, with or without constraints on the number of branch points, deal with *complete belief states* and hence no information about the world is lost. Although there is a close connection between the length of a plan and the information requirements for it, this dependency is not direct. If the agent keeps a history of the actions taken and the observations received, then the information requirement is proportional to the length of the plan. However, if the agent maintains a belief state, he does not need to store the history but to update the belief state at each time step and thus the requirement is $O(|D|2^{|D|})$ bits since this is the number of bits needed to store a belief (see below). In summary, neither the number of branch points in a plan nor its length determines the information requirements for the execution of the plan. Plans that constraint the amount of information usually take the form of finite-state controllers: the information requirement for a finite-state controllers is equal to the log of the number of states in the controller. Proposals that consider finite-state controllers for partially observable domains had been studied elsewhere [1,25,42,44,51].

Obviously, k-plans are acyclic plans that have a worst-case termination horizon. In this paper, we do not deal with cyclic or iterative plans that can be constructed either by using structured languages [39,40] or as functions that map states (or beliefs) into actions [2,6,15]. Cyclic plans are challenging to interpret as often it is not clear what is the role of the actions' non-deterministic effects [55]. Also, it is not completely clear how to define cyclic plans of bounded branching, specially for partially-observable problems.

3.1.1. Partially-observable domains

In partially-observable domains, the application of an operator o is accompanied by an observation z. This observation is the feedback that the agent receives from the environment, and depends on both the operator and the state that results after applying o. The feedback z is used by the agent to update its current belief b into the new belief $b_o^z \stackrel{\text{def}}{=} \{s \in b_o\colon\ s \vDash z\}$.

If the belief state b faithfully represents the possible current state of the system, then the set of observations that may be obtained after the application of o at b is the set $Z_b^o \stackrel{\text{def}}{=} \{z \in \text{Terms}(Z)\colon\ \exists s[s \in b_o \wedge s \vDash z]\}$ where $\text{Terms}(Z)$ denote all complete terms over the fluents Z. If there is more than one possible observation, the agent must consider more than one possible next belief state after the application of the operator and hence branching in belief space occurs.

Please cite this article in press as: B. Bonet, Conformant plans and beyond: Principles and complexity, Artificial Intelligence (2009), doi:10.1016/j.artint.2009.11.001

Just like the case of fully-observable problems, the execution of a linear plan $\pi = \langle o_1, \ldots, o_n \rangle$ may generate different trajectories. These trajectories are not over states but over belief states and thus we are forced to consider collections of belief states instead of collections of states. Let B_1 be a collection of belief states and define $B_{i+1} \stackrel{\text{def}}{=} Res(o_i, B_i)$, for $1 \leqslant i \leqslant n$, where

$$Res(o, B) \stackrel{\text{def}}{=} \begin{cases} \{b_0^z \colon b \in B,\ z \in Z_b^o\} & \text{if } o \text{ is applicable at each } b \in B, \\ \text{undefined} & \text{otherwise.} \end{cases}$$

The sequence π is said to be applicable on B_1 if no B_i is undefined. In this case, we let $Res(\pi, B_1) \stackrel{\text{def}}{=} B_{n+1}$.

If π is such that $Res(\pi, B)$ is a collection of goal belief states, we say that π is a 0-plan or linear plan for the collection B. A k-plan for a collection B is defined similarly as before: it is a labeled and directed tree $T = \langle V, E, r, \ell \rangle$ where r is the root node and ℓ is a labeling function that maps nodes n into sequence of operators $\ell(n)$ and edges e into belief states $\ell(e)$ such that:

- the height of T is less than or equal to k,
- for each node n, the function $\ell(n, \cdot)$ is one-to-one,
- the sequence of operators $\ell(r)$ is applicable at B,
- the collection of beliefs $\{\ell(r, n) \colon (r, n) \in E\}$ is equal to $Res(\ell(r), B)$, and
- for each edge (r, n), the subtree rooted at n is a $(k-1)$-plan for the singleton $\{\ell(r, n)\}$.

That is, the label of the root denotes the sequence of operators to apply at the collection B which results in the collection $Res(\ell(r), B)$. Then, after the application of the sequence, the agents calculate the current belief using all the information gathered during the application of the sequence (i.e. the feedback received by the agent). This belief b is a member of $Res(\ell(r), B)$. By the fourth condition, the tree contains a child n of r such that $\ell(r, n) = b$ and thus the subtree rooted at n is a $(k-1)$-plan for the singleton collection $\{b\}$. In particular, when n is a leaf of the tree, the sequence $\ell(n)$ is a linear plan for the singleton $\{b\}$.

These conditions characterize valid plans with at most k branch points for partially-observable problems: a plan with at most k branch points is a k-plan for the collection $\{b_I\}$. A plan with no bounds on branching for a partially-observable problem is a tree of arbitrary height, and a contingent plan is a plan with no bounds on branching in which each sequence $\ell(n)$ of operators contains only one operator.

4. Examples

We present two examples of partially-observable problems in which linear solutions are interesting and meaningful. The first problem is the game of Mastermind and the second one is related to a rover that must perform a number of tests in order to determine the presence of a chemical compound on the Martian surface.

4.1. The game of Mastermind

We provide a formalization for the game of Mastermind with 3 pegs and 3 colors; the case of more pegs and colors is similar. We use the propositions $\{s_{i,j} \colon 1 \leqslant i \leqslant 3,\ 1 \leqslant j \leqslant 3\}$ to denote the secret code with $s_{i,j}$ being true iff the peg i is of color j. For simplicity, we denote the number of exact and near matches of a guess *in unary* using the fluents $\{x_1, x_2, x_3\}$ and $\{n_1, n_2, n_3\}$ respectively. The propositions $s_{i,j}$ must be hidden as the codebreaker does not know their value while the other propositions must be observable as these provide the feedback for the guesses.

The initial situations correspond to all possible combinations of secret codes. Such combinations translate into all boolean assignments for the $s_{i,j}$ that assign a color for each peg and that forbid two different colors to be assigned to the same peg. For the observable propositions we assume, without loss of generality, that they are all false at the initial situation. Hence,

$$I \stackrel{\text{def}}{=} \bigwedge_{1 \leqslant i \leqslant 3} (s_{i,1} \vee s_{i,2} \vee s_{i,3}) \wedge \bigwedge_{1 \leqslant i \leqslant 3,\, 1 \leqslant j < k \leqslant 3} (\neg s_{i,j} \vee \neg s_{i,k}) \wedge \bigwedge_{1 \leqslant i \leqslant 3} (\neg x_i \wedge \neg n_i).$$

The goal of the game is to reach a 'state of knowledge' in which the secret code is known to the codebreaker. In the standard formulation of the game, the game ends when the codebreaker makes a guess that exactly matches the secret code. In our formulation however the game ends when the codebreaker discovers the secret code. These two formulations are not the same since it is possible to discover the secret code without making a guess that has 3 exact matches. In our formulation, the goal is

$$G \stackrel{\text{def}}{=} \bigwedge_{1 \leqslant i, j \leqslant 3} (\Box s_{i,j} \vee \Box \neg s_{i,j})$$

as it says that the truth-value for the $s_{i,j}$ is known to the codebreaker; i.e. the secret code is known.

Please cite this article in press as: B. Bonet, Conformant plans and beyond: Principles and complexity, Artificial Intelligence (2009), doi:10.1016/j.artint.2009.11.001

In the standard formulation, the goal can be defined as observing three exact matches that happens when the code-breaker makes a guess that exactly matches the secret code. However, such goal formula only works for computing contingent plans for Mastermind and not for computing linear plans.

Some problems involving knowledge can be translated into problems where the notion of knowledge is moved down to the syntactic level thus removing the need for a modal operator. These translations implement operators that manipulate the knowledge at the syntactic level using standard rules of inference such as the ones in the axiomatization **S5**. However, these approaches often increase the size of the translation by an exponential factor. Recently, Palacios and Geffner [48] gave sound and complete translations for computing conformant plans for unobservable problems into classical planning problems, but with a worst-case exponential increase in size.

We continue with the formulation of Mastermind for which it only remains to define the operators. Since the secret code does not change during the game, the operators do not modify it; they just return information via the observable fluents. In order to simplify the coding, we let fluents n_i to count the number of near plus exact matches in the guess. For example, if the code is $\langle 1, 2, 1\rangle$ and the guess is $\langle 2, 1, 1\rangle$, then the feedback is $(x, n) = (1, 3)$ as the first '1' and the '2' in the guess are near matches and the second '1' in the guess is an exact match. This variation of the game is equivalent to the original one because the amount of the information returned by each action is the same. Mastermind has 'guess(c_1, c_2, c_3)' for each guess of the form $\langle c_1, c_2, c_3\rangle$ with $c_i \in \{1, 2, 3\}$. The operators have empty precondition and the following conditional effects:

$$\left[\bigvee_{1\leqslant i\leqslant 3} s_{i,c_i} \rhd x_1\right] \wedge \left[\neg \bigvee_{1\leqslant i\leqslant 3} s_{i,c_i} \rhd \neg x_1\right] \wedge \left[\bigvee_{1\leqslant i<j\leqslant 3} s_{i,c_i} \wedge s_{j,c_j} \rhd x_2\right] \wedge \left[\neg \bigvee_{1\leqslant i<j\leqslant 3} s_{i,c_i} \wedge s_{j,c_j} \rhd \neg x_2\right]$$
$$\wedge \left[\bigwedge_{1\leqslant i\leqslant 3} s_{i,c_i} \rhd x_3\right] \wedge \left[\neg \bigwedge_{1\leqslant i\leqslant 3} s_{i,c_i} \rhd \neg x_3\right],$$
$$\left[\bigvee_{1\leqslant i,j\leqslant 3} s_{i,c_j} \rhd n_1\right] \wedge \left[\neg \bigvee_{1\leqslant i,j\leqslant 3} s_{i,c_j} \rhd \neg n_1\right] \wedge \left[\bigvee_{1\leqslant i<j\leqslant 3} \bigvee_{1\leqslant k,l\leqslant 3} s_{i,c_k} \wedge s_{j,c_l} \rhd n_2\right]$$
$$\wedge \left[\neg \bigvee_{1\leqslant i<j\leqslant 3} \bigvee_{1\leqslant k,l\leqslant 3} s_{i,c_k} \wedge s_{j,c_l} \rhd \neg n_2\right] \wedge \left[\bigwedge_{1\leqslant i\leqslant 3} \bigvee_{1\leqslant j\leqslant 3} s_{i,c_j} \rhd n_3\right]$$
$$\wedge \left[\neg \bigwedge_{1\leqslant i\leqslant 3} \bigvee_{1\leqslant j\leqslant 3} s_{i,c_j} \rhd \neg n_3\right].$$

The first six conditional effects compute the number of exact matches in unary, and the last six compute the number of exact plus near matches in unary.

4.2. The Mars Rover

We present an abstract formulation of an autonomous rover that must perform a set of experiments in order to ascertain the presence of some chemical compound on the Martian surface. The rover has no capabilities to perform in-depth analysis of samples. Thus an experiment consists in collecting a sample, performing some measurements on the sample, and transmitting back to Earth the results of the measurements.

The state of the system is specified using two different sets of fluents. First, feature fluents $\{f_1, \ldots, f_n\}$ that denote the presence or absence of n features on the surface of the planet, and second, power fluents $\{p_1, \ldots, p_m\}$ that denote the remaining power in the batteries of the rover. The power fluents measure the remaining power in unary; initially the rover has m power units and each experiment consumes one unit.

For simplicity, we assume that each experiment returns one bit of information whose truth-value only depends on the current but unknown state of the environment. Thus, we assume that for each experiment '$test_i$' there is a formula φ_i, over the feature fluents, that defines the information bit.

The Mars Rover problem is a partially-observable planning problem $P = \langle D, I, G, O, Z\rangle$ where

$$D = \{f_1, \ldots, f_n\} \cup \{p_1, \ldots, p_m\} \cup \{\text{bit}\},$$
$$I = \bigwedge_{1\leqslant i\leqslant m} p_i \wedge \neg\text{bit},$$
$$G = \bigwedge_{1\leqslant i\leqslant n} (\Box f_i \vee \Box\neg f_i),$$
$$O = \{\text{test}_1, \ldots, \text{test}_\ell\},$$
$$Z = \{\text{bit}\}$$

Please cite this article in press as: B. Bonet, Conformant plans and beyond: Principles and complexity, Artificial Intelligence (2009), doi:10.1016/j.artint.2009.11.001

in which the goal is to be certain about presence/absence of each feature f_i. The initial states correspond to all the 2^n interpretations for the features. The operators require and consume one unit of energy, and set or clear the information bit depending on whether the formula φ_i is true or false; i.e., for $1 \leqslant i \leqslant \ell$,

$$\text{test}_i = \left\langle p_1 \vee \cdots \vee p_m, (\varphi_i \rhd \text{bit}) \wedge (\neg\varphi_i \rhd \neg\text{bit}) \wedge \bigwedge_{1 \leqslant j \leqslant m} (\neg p_1 \wedge \cdots \wedge \neg p_{j-1} \wedge p_j \rhd \neg p_j) \right\rangle.$$

We consider two classes of solutions (plans) for the Mars Rover and how each class translates into assumptions on the capabilities of the rover. Solutions in the first class have no constraints on the number of branch points and hence they are contingent plans for the problem. This class corresponds to a rover that has sufficient capabilities to perform the filtering operation on belief states and thus able to update its belief state with the result of each test. In this setting, the plan is a contingent plan that generates a sequence of tests able to discover the exact configuration of features on the Martian surface.

Solutions in the second class have no branch points and thus are linear sequences of tests. This class corresponds to a rover that has no capabilities whatsoever to perform belief filtering. Therefore, the task of the rover is to perform all test in the sequence and then transmit the results back to Earth. The results are then analyzed by a team of human experts that determine, from the unique and fixed sequence of results, the exact configuration of features on the Martian surface.

In this example, the details about the capabilities of the rover are abstracted away from the representation of the problem, yet such details are not lost as they got *translated into requirements on the form of the solutions*. Thus, if the rover is able to process the results of the experiments, we look for contingent plans, but if the robot is unable to do so, we then look for linear plans.

Examples like this one show that the proposed language and novel planning problems are not only of pure theoretical interest, since they are related to very relevant and difficult real-life problems.

5. Complexity and decision problems

We only consider Turing Machines (TM) with semi-infinite tapes that halt on all inputs. We use DTM, NTM and ATM to denote deterministic, non-deterministic and alternating TMs respectively. We consider the standard complexity classes P (polynomial time), PSPACE (polynomial space), EXP (exponential time), EXPSPACE (exponential space), 2EXP (double-exponential time), 2EXPSPACE (double-exponential space), and their non-deterministic variants. The non-deterministic space classes are equal to the deterministic ones [56], and thus closed under complementation, e.g. NEXPSPACE = EXPSPACE = coEXPSPACE.

One can define the class of languages recognized by ATMs that start at an existential state, make at most $n-1$ alternations, and operate in polynomial time. Such class is denoted by Σ_n^P or Π_n^P if the initial state is universal. The union of all these classes form the Polynomial-Time Hierarchy (PH) [59] which can also be defined via oracles [62]. The following are standard facts [17,49,56]:

- $\text{P} \subseteq \Sigma_n^P \cup \Pi_n^P \subseteq \Sigma_m^P \cap \Pi_m^P \subseteq \text{PSPACE}$ for all $0 \leqslant n < m$ (hierarchy theorem).
- The following problem, called SAT_k, is complete for Σ_k^P: Given a quantified boolean formula (QBF) $\Psi = (\exists \bar{x}_1 \forall \bar{x}_2 \cdots Q\, \bar{x}_k)\Phi$, where Q is $\forall$ or $\exists$ whether k is even or odd. Is Ψ a valid formula?
- The following problem, called QBF, is complete for PSPACE: Given a QBF $\Psi = (\exists \bar{x}_1 \forall \bar{x}_2 \cdots Q\, \bar{x}_k)\Psi$, where Q is $\forall$ or $\exists$ whether k is even or odd. Is Ψ a valid formula?

Observe that k is a fixed constant in the problem SAT_k (not part of the input), while for QBF the number of quantifiers is variable. Thus, the problem QBF cannot be reduced to a unique SAT_k problem (unless PH collapses to such level).

5.1. Decision problems

The standard methodology is to cast decision problems for planning as languages. For example, the problem of deciding whether a given STRIPS planning problem has a valid plan correspond to the language

$$\text{PLAN-STRIPS} \stackrel{\text{def}}{=} \{\langle P\rangle\colon\ P \text{ is a STRIPS problem that has a plan}\}$$

where $\langle P\rangle$ is a suitable encoding of P. The length of P is defined as the length of its encoding. The following list contains the main decision problems considered in this article (we abbreviate 'fully-observable planning problem' as FOP and 'partially-observable planning problem' as POP):

- $\text{PLAN-ADL} \stackrel{\text{def}}{=} \{\langle P\rangle\colon\ P$ is an ADL planning problem (and hence FOP) that has a plan$\}$,
- $\text{PLAN-FO-CONT} \stackrel{\text{def}}{=} \{\langle P\rangle\colon\ P$ is a FOP that has a plan$\}$,
- $\text{PLAN-FO-LINEAR} \stackrel{\text{def}}{=} \{\langle P\rangle\colon\ P$ is a FOP that has linear plan$\}$,
- $\text{PLAN-FO-BRANCH}(k) \stackrel{\text{def}}{=} \{\langle P\rangle\colon\ P$ is a FOP that has a k-plan$\}$,

- PLAN-FO-BRANCH $\stackrel{\text{def}}{=} \{\langle P, k\rangle$: P is a FOP that has a k-plan and k is *written in binary*$\}$,
- PLAN-PO-CONT $\stackrel{\text{def}}{=} \{\langle P\rangle$: P is a POP hat has a plan$\}$,
- PLAN-PO-LINEAR $\stackrel{\text{def}}{=} \{\langle P\rangle$: P is a POP that has linear plan$\}$,
- PLAN-PO-BRANCH(k) $\stackrel{\text{def}}{=} \{\langle P\rangle$: P is a POP that has a k-plan$\}$,
- PLAN-PO-BRANCH $\stackrel{\text{def}}{=} \{\langle P, k\rangle$: P is a POP that has a k-plan and k is *written in binary*$\}$.

Observe that PLAN-FO-BRANCH(k) and PLAN-PO-BRANCH(k) represent collections of problems, one per each value of k. The problems PLAN-FO-BRANCH and PLAN-PO-BRANCH represent languages made of pairs $\langle P, k\rangle$ where P is a planning problem and k is an integer written in binary. PLAN-FO-BRANCH(k) and PLAN-FO-LINEAR can be reduced in a direct way to PLAN-FO-BRANCH, and thus their complexity is no more than the complexity of the latter. Similarly, for PLAN-PO-BRANCH(k), PLAN-PO-LINEAR and PLAN-PO-BRANCH.

Tight complexity bounds for PLAN-STRIPS, PLAN-ADL, PLAN-FO-CONT, PLAN-FO-LINEAR and PLAN-PO-CONT are known. Indeed, PLAN-STRIPS and PLAN-ADL are PSPACE-complete [11], PLAN-FO-CONT is EXP-complete [53],[1] PLAN-FO-LINEAR is EXPSPACE-complete [27], and PLAN-PO-CONT is 2EXP-complete [53] (the last two results are for problems with no modal formulae). The problems PLAN-FO-BRANCH(k), PLAN-FO-BRANCH, PLAN-PO-BRANCH(k), PLAN-PO-BRANCH and PLAN-PO-LINEAR are novel. The first two problems will be shown to be EXPSPACE-complete while the latter three will be shown to be 2EXPSPACE-complete.

As defined, PLAN-STRIPS, PLAN-ADL and PLAN-FO-CONT include problems with modal formulae. However, since in these problems the state of the system is known at each time point, the modalities play no role and they can be removed without changing the problems; thus, their complexity results remain valid. The results that need to be revised are those for PLAN-FO-LINEAR and PLAN-PO-CONT with modalities. As we will see, the complexity results of Haslum and Jonsson [27] and Rintanen [53] also remain valid for these classes.

6. Inclusion results

We first revise the complexity of PLAN-FO-LINEAR and PLAN-PO-CONT for problems with modal formulae, and then provide inclusion results for PLAN-FO-BRANCH(k), PLAN-FO-BRANCH, PLAN-PO-BRANCH(k), PLAN-PO-BRANCH and PLAN-PO-LINEAR.

Observe that with n propositional symbols, there are at most 2^n planning states and hence a set of states can be represented in exponential space with $n2^n$ bits, i.e. at most 2^n states each taking space n. Also, recall that the truth of a modal formula can be decided in polynomial space in the size of the formula.

The cases for PLAN-FO-LINEAR and PLAN-PO-CONT are essentially the same as when there are no modalities: the only change is that it may be necessary to test the validity of a modal formulae, yet this can be done within the resources used by the algorithm.

For PLAN-FO-LINEAR, the existence of a plan can be decided in non-deterministic exponential space with a TM that only stores a single belief state and a counter. The machine starts with the initial belief state and the counter initialized to zero. Then, the machine enters a loop in which non-deterministically chooses an applicable operator, applies it and cycles until the current belief becomes a goal belief or after reaching 2^{2^n} steps. Since a valid plan must have at most 2^{2^n} operators and checking the validity of modal formulae requires polynomial space, the TM correctly decides the existence of a plan and works in exponential space. Observe that the machine does not compute a plan, it just decides if a plan exists.

Theorem 1. PLAN-FO-LINEAR *is in EXPSPACE. Since completeness holds for the restricted case of problems without modalities,* PLAN-FO-LINEAR *is EXPSPACE-complete.*

PLAN-PO-CONT can be solved with an ATM using exponential space for which the accepting computation trees correspond to contingent plans (the proof is a special case of the proof of Theorem 3 given below). Since ATMs with exponential space bounds can be simulated with DTMs with double-exponential time bounds [13], we have

Theorem 2. PLAN-PO-CONT *is in 2EXP. Since completeness holds for the restricted case of problems without modalities,* PLAN-PO-CONT *is 2EXP-complete.*

The inclusion of PLAN-FO-BRANCH in EXPSPACE is shown with an ATM that makes at most k alternations. Note that each branch point in the plan corresponds to fully observing the current state of the system and then planning thereafter. Since a different plan must be found for each possible state, the branch can be simulated with a transition from a universal state. A final simulation of the ATM with a DTM shows the inclusion in EXPSPACE.

Theorem 3. PLAN-FO-BRANCH *is in EXPSPACE. Hence,* PLAN-FO-BRANCH(k) *is in EXPSPACE.*

[1] The complexity remains EXP-complete even for testing the existence of plans that reach the goal with probability $\geqslant t$ for probabilistic problems with full observability [41].

Please cite this article in press as: B. Bonet, Conformant plans and beyond: Principles and complexity, Artificial Intelligence (2009), doi:10.1016/j.artint.2009.11.001

Proof. Consider the input $\langle P, k\rangle$, where $P = \langle D, I, G, O\rangle$ is a fully-observable planning problem and k is a bound on the number of branch points written in binary. Let $|D| = n$. The following ATM decides if there is a k-plan for P:

M: on input $\langle P, k\rangle$
1. $K := k$; $b := \{s\colon\ I \vDash s\}$; $steps := 0$;
2. **if** $b \vDash G$ **then** ACCEPT;
3. ∃-branch: **choose** either APPLY or BRANCH;
4. **if** BRANCH **then**
5. **if** $K = 0$ **then** REJECT;
6. $K := K - 1$;
7. ∀-branch: **for each** $s \in b$ **do** $b := \{s\}$;
8. **goto** 2;
9. **else if** APPLY **then**
10. **if** $steps = 2^{2^n}$ **then** REJECT;
11. ∃-branch: **choose** operator $a = \langle \varphi, \alpha\rangle$ such that $b \vDash \varphi$;
12. $b := Res(a, b)$;
13. $steps := steps + 1$;
14. **goto** 2;
15. **end**

The ATM is in EXPSPACE since there are at most 2^n states so a subset of states can be stored in $O(n2^n)$ bits, the counters *steps* and K require $O(2^n + \log K)$ bits, and the tests in lines 2 and 11 can be done in polynomial space. Use now Borodin's Theorem (Appendix A) to conclude that M can be transformed into a DTM with an exponential space bound. □

The same idea works for the inclusion of PLAN-PO-BRANCH in 2EXPSPACE except that this time collections of belief states are stored instead of belief states, and thus the ATM requires $O(n2^n 2^{2^n})$ bits.

Theorem 4. PLAN-PO-BRANCH *is in 2EXPSPACE. Hence,* PLAN-PO-BRANCH(k) *and* PLAN-PO-LINEAR *are in 2EXPSPACE.*

Proof. Consider the input $\langle P, k\rangle$, where $P = \langle D, I, G, O, Z\rangle$ is a partially-observable planning problem and k is a bound on the number of branch points written in binary. The following ATM decides if there is a k-plan for P:

M: on input $\langle P, k\rangle$
1. $K := k$; $B := \{\{s\colon\ I \vDash s\}\}$; $steps := 0$;
2. **if** $b \vDash G$ for all $b \in B$ **then** ACCEPT;
3. ∃-branch: **choose** either APPLY or BRANCH;
4. **if** BRANCH **then**
5. **if** $K = 0$ **then** REJECT;
6. $K := K - 1$;
7. ∀-branch: **for each** $b \in B$ **do** $B := \{b\}$;
8. **goto** 2.
9. **else if** APPLY **then**
10. **if** $steps = 2^{2^{2^n}}$ **then** REJECT;
11. ∃-branch: **choose** operator $a = \langle \varphi, \alpha\rangle$ such that $b \vDash \varphi$ for all $b \in B$;
12. $B := Res(a, B)$;
13. $steps := steps + 1$;
14. **goto** 2.
15. **end**

Since there are at most 2^n states and 2^{2^n} subsets of states, a collection B of belief states can be stored in $n2^n 2^{2^n}$ bits and the ATM has a double-exponential space bound. As before, Borodin's Theorem implies that M can be transformed into a DTM with a double-exponential space bound. □

For hardness results, we follow the work of Haslum and Jonsson [27] and use regular expressions with exponentiation and the corresponding non-deterministic finite automata with counters. Automata provide a fruitful approach for hardness results as it is easy to simulate their behavior with planning problems.

7. Regular expressions and automata

Regular expressions with exponentiation (REE) extend regular expressions with an operation of exponentiation of the form $\alpha \uparrow k$ where α is a regular expression (with exponentiation) and k is a positive integer *written in binary*. The expression

Please cite this article in press as: B. Bonet, Conformant plans and beyond: Principles and complexity, Artificial Intelligence (2009), doi:10.1016/j.artint.2009.11.001

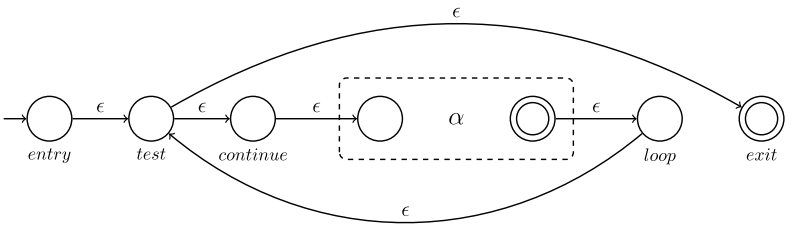

Fig. 2. Non-deterministic finite automaton with counters for $\alpha \uparrow k$.

$\alpha \uparrow k$ denotes the language $\alpha \cdots \alpha$ where α is repeated k times. A REE can be transformed into a regular expression without exponentiation by replacing each subexpression $\alpha \uparrow k$ by the concatenation of k copies of α. This transformation however increases the length of the expression exponentially; e.g. the length of $\alpha \uparrow k$ is $1 + |\alpha|\lceil \log k \rceil$ while the length of $\alpha \cdots \alpha$ is $k|\alpha|$. As usual, the length $|\alpha|$ is defined as the number of symbols in it.

Regular expressions with exponentiation are typically used to establish intractability results for EXPSPACE and beyond. The main tool utilized is the following classical result about REEs; we refer to the version that appears in Sipser's book [56], yet also consult Hopcroft and Ullman [32] or the source [46].

Theorem 5. *Let α be a REE over alphabet Σ. Checking whether $\alpha = \Sigma^*$ is EXPSPACE-hard.*

Proof sketch. Given a language $A \in$ EXPSPACE decided by a DTM with an exponential-space bound $s(n) = 2^{n^k}$, we show a polynomial-time reduction f that on input ω of length n produces a REE $f(\omega) = \alpha$ such that M accepts ω iff $\alpha = \Sigma^*$ where the alphabet Σ only depends on M and $|\alpha|$ is polynomial in n.

The idea is to let α denote all invalid or non-rejecting computation histories of M on ω. A rejecting computation history of M on ω is a sequence of configurations (snapshots) of M in which the first configuration is the initial configuration of M on ω, the last configuration is a rejecting configuration, and each configuration in the sequence follows from the preceding one from a single step of M. The REE α is defined to denote all strings that either do not begin with the initial configuration of M on ω, do not contain a rejecting configuration, or have a configuration that does not follow from the preceding one [56, pp. 344–347]. The size of α not counting the exponents is $O(n \log n)$, and counting the exponents is $O(n^k)$ since all exponents in α are in the set $\{1, \ldots, n+1, 2^{n^k} - n - 2, 2^{n^k} - 2, 2^{n^k}\}$. By construction, M accepts ω iff there is no rejecting computation history of M on ω (since M is deterministic) iff α denotes all strings over its alphabet. □

The computational models that correspond to regular expressions are deterministic and non-deterministic finite automata. In the presence of exponents, the automata need to be extended with counters. We define a class of non-deterministic finite automata with counters (NFACs) that is simple yet sufficient for our needs; we could use the more general NFACs of Haslum and Jonsson at the cost of getting more complex proofs.

An NFAC is a non-deterministic finite automaton augmented with a set C of *counters of bounded capacity*. Each counter $c \in C$ corresponds to a sub-expression of the form $\alpha \uparrow k$ that is implemented with five states $entry_c$, $test_c$, $continue_c$, $exit_c$ and $loop_c$, and a capacity bound $bound_c$. These elements are specific to the counter and thus are subscripted accordingly. We call the states in $\{entry_c, test_c, continue_c, loop_c, exit_c\colon\ c \in C\}$ as cstates and denote the latter set as *cstates*.

The automaton for counter c associated with $\alpha \uparrow k$ is shown in Fig. 2. The operation of the automaton is as follows. Initially, the counter is set to zero. Upon visiting $entry_c$ the counter is initialized to $bound_c$ and an ϵ-transition is made to $test_c$; this is the only transition from $entry_c$. In $test_c$, the machine makes an ϵ-transition to either $continue_c$ or $exit_c$ depending on whether $c > 0$ or not; these are the only transitions from $test_c$ and into $continue_c$ and $exit_c$. In $continue_c$, the machine makes an ϵ-transition to the initial state of α, and an ϵ-transition from the final state of α is made into $loop_c$. Upon visiting $loop_c$, the machine decrements the counter c and makes an ϵ-transition to $test_c$; the only transitions to $test_c$ are either from $entry_c$ or $loop_c$.

Formally, an NFAC $M = \langle Q, \Sigma, \delta, q_0, F, C \rangle$ consists of a set Q of states, an input alphabet Σ, a transition function $\delta : Q \times \Sigma \to 2^Q$, an initial state $q_0 \in Q$, a subset $F \subseteq Q$ of accepting (final) states, and a set of counters C. Each counter has five states associated with it as described above. The language $L(M)$ recognized by M is the set of words in Σ^* that induce a path from the initial state q_0 into an accepting state, the size of M is $|M| \stackrel{\text{def}}{=} |\delta| + \sum_{c \in C} \lceil \log bound_c \rceil$, and the relation between REEs and NFACs is the following.

Theorem 6. *For each REE α there is an NFAC M_α of polynomial size in $|\alpha|$ such that $\omega \in \alpha$ iff $\omega \in L(M_\alpha)$. Conversely, for each NFAC M there is a REE α_M of polynomial size in $|M|$ such that $\omega \in L(M)$ iff $\omega \in \alpha_M$.*

Proof. Direct. Left to the reader. □

Please cite this article in press as: B. Bonet, Conformant plans and beyond: Principles and complexity, Artificial Intelligence (2009), doi:10.1016/j.artint.2009.11.001

From now on we assume complete automata. An automaton M is complete if $\delta(q,a)$ is non-empty for all $q \in Q$ and $a \in \Sigma \cup \{\epsilon\}$. A non-complete automaton can be easily transformed into a complete one by adding a non-accepting state q_{sink} such that $\delta(q,a) = \{q_{sink}\}$ for all (q,a) with $\delta(q,a) = \emptyset$, and $\delta(q_{sink},a) = \{q_{sink}\}$ for all $a \in \Sigma \cup \{\epsilon\}$. Furthermore, we assume without loss of generality, that there is exactly one transition from the states $exit_c$ into non-sink states which is labeled with ϵ.

A configuration (snapshot) for M is a pair $\theta = \langle q, \nu \rangle$ such that $q \in Q$ specifies the current state of M and $\nu : C \to \mathbb{N}$ specifies the current value for each counter in M. A configuration is accepting if its state is accepting. The initial configuration of the automaton is the pair $\theta_0 = \langle q_0, \nu_0 \rangle$ where $\nu_0(c) = 0$ for all $c \in C$.

The transition function δ can be extended into a function $\hat{\delta}$ that maps configurations and words in Σ^* into subsets of configurations such that $\theta' \in \hat{\delta}(\theta, \omega)$ iff the configuration θ' can be reached from θ through a path labeled with ω (perhaps including ϵ-transitions) (the details are in Appendix B). Thus, $L(M) \stackrel{\text{def}}{=} \{\omega \in \Sigma^* \colon\ \hat{\delta}(\theta_0, \omega)$ contains an accepting configuration$\}$.

7.1. The proof of Haslum and Jonsson

The EXPSPACE-hardness for conformant planning of fully-observable problems is established with a reduction from the problem of checking $\alpha \neq \Sigma^*$ to the problem of checking if a fully-observable planning problem has a conformant plan, and then using the fact that EXPSPACE is closed under complementation.

Indeed, let α be a REE over alphabet Σ and $M = \langle Q, \Sigma, \delta, q_0, F, C \rangle$ an NFAC that recognizes α. The idea is to construct a planning problem P that *simulates* the extended transition function $\hat{\delta}$. The states of the planning problem represent the configurations of M and thus can be denoted by $[q, \nu]$ for configuration $\langle q, \nu \rangle$. The initial belief for P corresponds to the singleton $\{[q_0, \nu_0]\}$, and there are (non-deterministic) operators o_a for each $a \in \Sigma \cup \{\epsilon\}$. The construction guarantees that for every pair of configurations $\langle q, \nu \rangle$ and $\langle q', \nu' \rangle$,

$$\big[q', \nu'\big] \in Res\big(\pi_{pad} o_{a_1} \pi_{pad} o_{a_2} \pi_{pad} \cdots \pi_{pad} o_{a_n} \pi_{pad}, \big\{[q, \nu]\big\}\big) \quad \text{iff } \big\langle q', \nu' \big\rangle \in \hat{\delta}\big(\langle q, \nu \rangle, a_1 \cdots a_n\big),$$

where π_{pad} is a fixed sequence of o_ϵ operators.

If the goal is defined as being in a non-accepting configuration, then P has a valid linear plan iff $\alpha \neq \Sigma^*$. The key issue for constructing P is that a counter of exponential capacity requires a polynomial number of bits and thus can be represented with a polynomial number of fluents.

In our case, we aim to show that deciding the existence of a linear plan for a partially-observable problems is 2EXPSPACE-hard, yet a direct simulation of NFACs with counters of double-exponential capacity does not work since it requires a planning problem with an exponential number of fluents. However, more compact encodings of counters can be achieved when collections of belief states instead of just belief states are considered.

8. Compact representation of counters

Let c be a counter of exponential size, and hence of double-exponential capacity, with bits c_i for $0 \leqslant i < 2^n$, and let P be a planning problem with n propositional "markers" $m_{c,k}$ for $0 \leqslant k < n$ (often simply denoted by m_k when c is clear from context). We will show how to encode a value for c as a collection of subsets of markers.

To illustrate the idea, observe that a value for c can be represented as the collection of *positions* for the 1-bits in the binary expansion of c; i.e. by the set $\{i \colon\ c_i = 1\}$. For example, the value $72 = 01001000_{\text{bin}}$ is represented by the set $\{3, 6\}$ since the 1-bits in the binary expansion of 72 are the third and the sixth bit (the least-significant, the rightmost, is the zeroth bit). This representation cannot be used directly since there is an exponential number of positions. However, if each position is further represented in binary using the markers, then the value of a counter can be represented with a collection of subsets of markers. In the example, the value 72 is represented by the collection $\{\{m_0, m_1\}, \{m_1, m_2\}\}$ since $3 = 2^0 + 2^1$ and $6 = 2^1 + 2^2$. In general, the value of a counter is represented by the following collection of subsets of markers:

$$b_c = \big\{\big\{m_{c,k} \colon\ (k\text{th bit in } i_{\text{bin}}) = 1\big\} \colon\ c_i = 1\big\}.$$

For example, the collection $\{\emptyset\}$ represents the value $c = 1 = 00000001_{\text{bin}}$ since the unique 1-bit is in position 0, $\{\{m_0\}, \{m_2\}\}$ represents the value $c = 18 = 00010010_{\text{bin}}$ since the positions of the 1-bits are $\{1, 4\}$ that written in binary are $\{001_{\text{bin}}, 100_{\text{bin}}\}$, and $\{\emptyset, \{m_0, m_1, m_2\}\}$ represents the value $c = 129 = 10000001_{\text{bin}}$ since the positions of the 1-bits are $\{0, 7\} = \{000_{\text{bin}}, 111_{\text{bin}}\}$.

In addition to the propositional markers, we use a proposition 'A_c' that indicates whether a subset of markers is active for counter c in a given representation; when c is clear from context, we just write A. Non-active subsets are ignored when performing operations on c yet their existence alleviates the complexity of some operations. Hence, for example, the value $c = 72$ is represented by the belief $\{\{m_0, m_1, A\}, \{m_1, m_2, A\}\}$. From now on, we use the notation $c \sim b$ to denote that the counter c is represented by the belief state b.

8.1. Arithmetic on counters

In order to simulate NFACs with planning problems, we need to perform simple arithmetics on counters such as comparisons and decrement operations.

Please cite this article in press as: B. Bonet, Conformant plans and beyond: Principles and complexity, Artificial Intelligence (2009), doi:10.1016/j.artint.2009.11.001

Let c be a counter and b a belief state such that $c \sim b$. Then, $c = 0$ iff there is no active state in b, and $c > 0$ iff there is at least one active state. These two conditions are written in propositional modal logic as $\neg A_c$ and $\Diamond A_c$ which are abbreviated as '$c = 0$' and '$c > 0$' respectively. Notice that A_c cannot be used to test $c > 0$ since the former holds iff each state in b is active. As we will see, the decrement operation may render some active states $s \in b$ inactive.

The decrement operation must subtract one from the value of the counter. For example, it must change the value $72 = 01001000_{\text{bin}}$ represented by $b_{72} = \{\{m_0, m_1, A\}, \{m_1, m_2, A\}\}$ into the value $71 = 01000111_{\text{bin}}$ represented by $b_{71} = \{\{A\}, \{m_0, A\}, \{m_1, A\}, \{m_1, m_2, A\}\}$. That is, the subset $\{m_0, m_1, A\}$ needs to be replaced by the subsets $\{A\}$, $\{m_0, A\}$ and $\{m_1, A\}$. The general principle involved is to replace the subset for the *least-significant 1-bit* (the first 1 from right to left in c's binary expansion) with the subsets that represent all bits of less significant value. We thus need to first identify the subset to replace and then to generate the replacement subsets.

For the first task, we consider a sequence of effects that progressively isolate the active subsets that represent the least-significant 1-bit. At the beginning, all active subsets are flagged with the proposition '$\min_c$' using the effect $A_c \rhd \min_c$. Then, for $k = n$ to $k = 0$, effects that filter out flagged subsets that are dominated with respect to the m_k marker are applied in succession. These effects have the form

$$\begin{aligned}
&\mathit{flag}_c \stackrel{\text{def}}{=} A_c \rhd \min_c,\\
&\mathit{filter}_{c,n-1} \stackrel{\text{def}}{=} \big(\min_c \wedge m_{c,n-1} \wedge \Diamond\big(\min_c \wedge \neg m_{c,n-1}\big)\big) \rhd \neg \min_c,\\
&\quad\vdots\\
&\mathit{filter}_{c,0} \stackrel{\text{def}}{=} \big(\min_c \wedge m_{c,0} \wedge \Diamond\big(\min_c \wedge \neg m_{c,0}\big)\big) \rhd \neg \min_c\,.
\end{aligned}$$

In the example, if we apply the succession of effects $\langle \mathit{flag}, \mathit{filter}_2, \mathit{filter}_1, \mathit{filter}_0\rangle$ to b_{72}, we obtain the belief $\{\{m_0, m_1, A, \min\}, \{m_1, m_2, A\}\}$ in which the subset to replace is the one flagged with min.

Once the subset to replace is identified, the replacements are generated by processing each marker in parallel. In the example, the least-significant corresponds to $\{m_0, m_1, A\}$. The decrement operation must clear this bit and turn on the 2nd, 1st and 0th bits since $72 = 01001000_{\text{bin}}$ is converted into $71 = 01000111_{\text{bin}}$. Thus, this state must be replaced by $\{A\}$, $\{m_0, A\}$ and $\{m_1, A\}$.

The replacement effect processes each marker m_k *non-deterministically* from $k = 0$ to $k = n$. The non-determinism is utilized to implement the parallel execution of individual effects, one per marker, whose results are then joined together. The effects are defined as:

- clear the flagged status,
- if no marker holds, then invalidate the state,
- if m_k holds, then clear m_k and non-deterministically flip each marker m_j for $0 \leqslant j < k$.

In the example, the state $s = \{m_0, m_1, A, \min\}$ needs to be replaced. Since m_2 does not hold in s, its processing generates nothing. The processing of m_1 clears m_1 and flips m_0 generating $\{A\}$ and $\{m_0, A\}$, The processing of m_0 clears m_0 and flips nothing generating $\{m_1, A\}$. Therefore, the overall effect is the collection $\{\{A\}, \{m_0, A\}, \{m_1, A\}\}$.

Let us illustrate this further with a decrement on $b_{71} = \{\{A\}, \{m_0, A\}, \{m_1, A\}, \{m_1, m_2, A\}\}$. After the identification effects, the belief is transformed into $\{\{A, \min\}, \{m_0, A\}, \{m_1, A\}, \{m_1, m_2, A\}\}$ so the subset to replace is $\{A, \min\}$ which corresponds to the least-significant 1-bit. Because this state satisfies no marker, the effect of the replacement is to invalidate the state by clearing the fluent A. Hence, the belief transforms into $\{\emptyset, \{m_0, A\}, \{m_1, A\}, \{m_1, m_2, A\}\}$ that represents $70 = 01000110_{\text{bin}}$ since the empty subset is non-active.

The decrement operation is implemented by the following sequence of $n + 2$ operators

$$\text{decrement}(c) \stackrel{\text{def}}{=} \big\langle \langle \top, \mathit{flag}_c\rangle, \langle \top, \mathit{filter}_{c,n-1}\rangle, \ldots, \langle \top, \mathit{filter}_{c,0}\rangle, \langle \top, \mathit{replace}_c\rangle \big\rangle,$$

where the $\mathit{replace}_c$ effect is

$$\neg \min_c \wedge \left[\left(\min_c \wedge \bigwedge_{k=0}^{n-1} \neg m_{c,k}\right) \rhd \neg A_c\right] \wedge \left[\left(\min_c \wedge \bigvee_{k=0}^{n-1} m_{c,k}\right)\right] \rhd \bigoplus_{k=0}^{n-1} \left[\left(m_{c,k} \rhd \neg m_{c,k} \wedge \bigwedge_{j=0}^{k-1} (m_{c,j} \oplus \neg m_{c,j})\right)\right].$$

The subsequence of the first $n + 1$ operators is called the identification stage and the last operator is called the replacement stage. The correctness of the arithmetic operations are summarized in the following result.

Theorem 7. *Let c be a counter and b a belief state such that $c \sim b$. Then,*

(a) $c = 0$ *iff* $b \vDash \neg A_c$,
(b) $c = 1$ *iff* $b \vDash \Diamond A_c \wedge (A_c \rightarrow \bigwedge_{k=0}^{n} \neg m_{c,k})$,

Please cite this article in press as: B. Bonet, Conformant plans and beyond: Principles and complexity, Artificial Intelligence (2009), doi:10.1016/j.artint.2009.11.001

(c) $c > 1$ *iff* $b \models \Diamond(A_c \wedge \bigvee_{k=0}^{n} m_{c,k})$,
(d) *if* $c = 0$, *then* $c \sim \mathit{Res}(\text{decrement}(c), b)$,
(e) *if* $c > 0$, *then* $c - 1 \sim \mathit{Res}(\text{decrement}(c), b)$.

Proof. Claims (a)–(c) are direct. For (d), if $c = 0$ then no state is flagged since none is active. Therefore, $\mathit{replace}_c$ has no effect on b.

For (e), we first show that the identification stage is correct. That is, the states that remain flagged after the identification correspond to the least-significant 1-bit of c. The correctness of the identification is a direct consequence of the following facts:

- *After applying* flag_c, *all active states are flagged.*
- *After applying* $\mathit{filter}_{c,k}$, *all flagged states agree on the interpretation of markers* $m_{c,\ell}$ *for* $\ell \geqslant k$. This is true for $k = n - 1$. Assume that is true for $k + 1$. Then, after $\mathit{filter}_{c,k}$ is applied, all resulting states satisfy $m_{c,k}$ or all resulting states satisfy $\neg m_{c,k}$.
- *At the end, all flagged states agree on the interpretation of all markers.* Direct by previous fact.
- *At the end, there is at least one flagged state.* Since $c > 0$, there is some active state that is initially flagged. Suppose that there are no flagged states at the end, and let k be the integer such that the input belief b to $\mathit{filter}_{c,k}$ has some flagged states, and its output has none. Then, it must be the case that for each flagged $s \in b$ there is a flagged $s' \in b$ such that $m_{c,k} \in s$ and $m_{c,k} \notin s'$. But then, $\mathit{filter}_{c,k}$ cannot clear the flag of s' contradicting the choice of k.
- *There is a flagged state that corresponds to the least-significant* 1*-bit.* Suppose this is not true. Then, since the state s for the least-significant 1-bit is initially flagged, there is k such that $\mathit{filter}_{c,k}$ clears the flag in s, and thus there is a flagged s' such that $m_{c,k} \in s$ and $m_{c,k} \notin s'$. Since all states before applying $\mathit{filter}_{c,k}$ agree on the interpretation of markers $m_{c,\ell}$, for $\ell > k$, then s' corresponds to a less significant 1-bit than s contradicting the choice of s.

We now show that the replacement stage is correct. Let $b = \{s_1, \ldots, s_p, s\}$ be the input belief to $\mathit{replace}_c$ where s is the flagged state that represents the least-significant 1-bit. We must show that $\mathit{replace}_c$ generates a belief $\{s_1, \ldots, s_p, s'_1, \ldots, s'_q\}$ in which the states s'_i represent all bits of lesser significance than s. We consider two cases whether s represents the 0th bit or other bit. In the first case, s has no markers. Then, $\mathit{replace}_c$ clears the active fluent and thus decrements the counter by one unit.

Suppose now that s contains the markers $\{m_{i_1}, \ldots, m_{i_\ell}\}$ for some $0 \leqslant i_1 < \cdots < i_\ell < n$. The bits of less significance than s are in correspondence with the subsets in

$$\bigcup_{k=1}^{\ell} \big\{ \{m_{i_{k+1}}, \ldots, m_{i_\ell}\} \cup C\colon\ C \in \wp\big(\{m_0, m_1, \ldots, m_{i_k - 1}\}\big) \big\}, \tag{1}$$

where $\wp(A)$ denotes the power set of A. We show that $\mathit{replace}_c$ generates these subsets only. Since s satisfies some marker, we only need to consider the second conditional effect. Notice that for any subset A the effect $\bigwedge_{a \in A}(a \oplus \neg a)$ generates $\wp(A)$. Therefore, $\bigwedge_{j=0}^{i_k - 1}(m_j \oplus \neg m_j)$ generates $\wp(\{m_0, \ldots, m_{i_k - 1}\})$ and thus $\neg m_{i_k} \wedge \bigwedge_{j=0}^{i_k - 1}(m_j \oplus \neg m_j)$ generates $\{\{m_{i_{k+1}}, \ldots, m_\ell\} \cup C\colon\ C \in \wp(\{m_0, \ldots, m_{i_k - 1}\})\}$ when applied at s. Finish by observing that the second conditional effect translates into $\bigoplus_{k=1}^{\ell}[\neg m_{i_k} \wedge \bigwedge_{j=0}^{i_k - 1}(m_j \oplus \neg m_j)]$ when applied at s. □

The problem of identifying the subset of markers that correspond to the least-significant bit is related to the problem of selecting the preferred state in the belief state with respect to a lexicographic preference order defined with the markers m_i: a state s is preferred over state s' iff there is k such that s makes true m_k and s' does not, and s and s' agree on the interpretation of all markers m_ℓ for $\ell > k$. There is no modality-free formula φ that selects the preferred state in a belief state, i.e. $s \models \varphi$ iff s is preferred in b, nor we know of a short (polynomially long) modal formula able to do so. This is the main reason for performing the identification stage during the decrement operation on counters.

We finish this section by noticing that multiple counters can be simultaneously encoded as cross-products of subsets of markers. Since the markers, the active fluents and the min fluents are all tagged with the counters, we can perform arithmetic operations on different counters simultaneously.

The compact encoding of counters and their arithmetic operations are the main tools needed to show the lower bounds on the complexity of decision problems.

9. Hardness results

The EXPSPACE-hardness of PLAN-FO-BRANCH(k) is shown by giving polynomial-time reductions from PLAN-FO-BRANCH(k) to PLAN-FO-BRANCH($k + 1$) for all $k \geqslant 0$. Since PLAN-FO-BRANCH(0) is EXPSPACE-hard [27], then PLAN-FO-BRANCH(k) is also EXPSPACE-hard for all $k \geqslant 0$. This result together with the inclusion results establish exact complexity bounds for PLAN-FO-BRANCH(k).

Please cite this article in press as: B. Bonet, Conformant plans and beyond: Principles and complexity, Artificial Intelligence (2009), doi:10.1016/j.artint.2009.11.001

Theorem 8. *For $k \geqslant 0$,* PLAN-FO-BRANCH(k) *is polynomial-time reducible to* PLAN-FO-BRANCH($k+1$). *Therefore,* PLAN-FO-BRANCH(k) *and* PLAN-FO-BRANCH *are EXPSPACE-complete.*

Proof. Let $P = \langle D, I, G, O\rangle$ be a planning problem with full observability. We need to construct a fully-observable problem $P' = \langle D', I', G', O'\rangle$ such that P has a k-plan iff P' has a $(k+1)$-plan. The problem P' is defined as follows:

$$D' = D \cup \{p, q_1, q_2, r\},$$
$$I' = I \wedge \neg p \wedge \neg q_1 \wedge \neg q_2 \wedge \neg r,$$
$$G' = r,$$
$$O' = \big\{\langle \neg p \wedge \varphi, \alpha\rangle\colon\ \langle \varphi, \alpha\rangle \in O\big\} \cup \big\{o_1 = \big\langle \neg p \wedge G,\ p \wedge (q_1 \oplus q_2)\big\rangle, o_2 = \langle p \wedge q_1, r\rangle, o_3 = \langle p \wedge q_2, r\rangle\big\},$$

where p, q_1, q_2 and r are new propositional symbols, and o_1, o_2 and o_3 are new operators. Clearly, the problem P' can be constructed from P in polynomial time.

We claim that P has a k-plan iff P' has a $(k+1)$-plan. Indeed, let π be a k-plan for P. That is, π makes at most k observations along each branch, and each branch finishes at a state that satisfies G. If at the end of each branch, we apply operator o_1 followed with an observation, to separate the states that satisfy q_1 from the ones that satisfy q_2, and finally apply o_2 and o_3 depending on whether q_1 or q_2 is satisfied, then we have a $(k+1)$-plan for P'.

Conversely, let π' be a $(k+1)$-plan for P'. The last operator along a branch in π' must be o_2 or o_3. Thus, the second to last operator must be o_1. After o_1 is applied, there are states that satisfy q_1 and states that satisfy q_2. Therefore, π' must make a branch (observation) after the application of o_1 in order to separate these states. Since the precondition of o_1 includes the goal G of P, then the branches of π' up to the application of o_1 form k-plan for P. □

We now show the 2EXPSPACE-hardness result for partially-observable domains. The proof idea is similar to the one used by Haslum and Jonsson except that we need to deal with counters of double-exponential capacity. Let us first revise the proof of Theorem 5 (cf., e.g., [56]). Given a DTM M with an exponential-space bound $e(n) = 2^{p(n)}$, with $p(n) = n^k$, and a word $\omega \in \Sigma^*$ of size n, let $\alpha = \alpha(M, \omega)$ be a REE of polynomial length in $|M| + n$ such that M accepts ω iff $\alpha = \Sigma^*$. If the space bound is replaced with a double-exponential space bound, α remains the same except that the exponents in α change from exponential to double-exponential. Indeed, for the space bound $e(n)$, the exponents in α are among $\{1, \ldots, n+1, 2^{n^k} - n - 2, 2^{n^k} - 2, 2^{n^k}\}$, while for a space bound of the form $d(n) = 2^{2^{p(n)}}$, the exponents are among $\{1, \ldots, n+1, 2^{2^{n^k}} - n - 2, 2^{2^{n^k}} - 2, 2^{2^{n^k}}\}$. Thus, for double-exponential space bounds, counters of double-exponential capacity must be used.

Theorem 9. PLAN-PO-LINEAR *is 2EXPSPACE-hard. Hence,* PLAN-PO-LINEAR *and* PLAN-PO-BRANCH *are 2EXPSPACE-complete.*

Proof. Let M be a DTM with a double-exponential space bound $d(n) = 2^{2^{p(n)}}$, where $p(n) = n^k$ is a polynomial in n, and $\omega \in \Sigma^*$ be an input word for M of length n. Consider the REE $\alpha = \alpha(M, \omega)$ given in the proof of Theorem 5 but for the space bound $d(n)$. The size of α is exponential yet, if the sizes of the exponents are not accounted for, $|\alpha|$ is polynomial in $|M| + n$. Let $M_\alpha = \langle Q, \Sigma, \delta, q_0, F, C\rangle$ be the NFAC for α in which $|\delta|$ is polynomial in $|M| + n$ while $\sum_{c \in C} \lceil \log bound_c \rceil$ is exponential. We are going to construct, *in polynomial time*, a planning problem P with partial observability such that its linear plans simulate the NFAC. The main idea is to *encode the configurations of M_α as the belief states of P* as described in Section 8. The construction will guarantee that $L(M_\alpha) \neq \Sigma^*$ iff P has a valid linear plan. Therefore, M rejects ω iff $L(M_\alpha) \neq \Sigma^*$ iff P has a valid linear plan. Since 2EXPSPACE is closed under complementation, then PLAN-PO-LINEAR is 2EXPSPACE-hard.

From the NFAC $M_\alpha = \langle Q, \Sigma, \delta, q_0, F, C\rangle$, define the planning problem $P = \langle D, I, G, O, Z\rangle$ as follows. The fluents in P is the collection of symbols for:

- representing the state of the machine: $\{q\colon\ q \in Q\}$,
- representing the value of counters: $\{A_c, \min_c, m_{c,k}\colon\ c \in C,\ 0 \leqslant k \leqslant p(n)\}$, and
- the execution mode: $\{normal, init_c, dec_{c,k}, flag_c, filter_{c,\ell}, replace_c\colon\ c \in C,\ 0 \leqslant k \leqslant n+2,\ 0 \leqslant \ell \leqslant p(n)\}$.

The observable symbols are just the symbols for the state of the machine, i.e. $Z = \{q\colon\ q \in Q\}$. The reason for this is that since belief states represent configurations of the machine, each belief state must make one and only one such symbol true. Making them observable, forces the beliefs in the collection B to satisfy this property. Each symbol has a precise role which will become clear as we describe the construction of P.

The descriptions I and G reflect the initial and non-accepting configurations respectively:

$$I \stackrel{\text{def}}{=} q_0 \wedge normal \wedge \bigwedge \big\{\neg p\colon\ p \in D \setminus \{q_0, normal\}\big\},$$
$$G \stackrel{\text{def}}{=} normal \wedge \bigvee_{q \notin F} q.$$

Please cite this article in press as: B. Bonet, Conformant plans and beyond: Principles and complexity, Artificial Intelligence (2009), doi:10.1016/j.artint.2009.11.001

Formula I encodes the initial configuration of M_α with state q_0 and the value of all counters set to 0 (since all active fluents are off). Formula G encodes all non-accepting configurations in which the state of M_α is not in F. The symbol *normal* is used to express that the current belief states *faithfully represent* the possible configurations of M_α. It is turned off whenever some transformation involving more than one operator, e.g. when decreasing the value of a counter, is being applied, and restored when the transformation finishes.

The planning operators are divided in two classes $O = \{o_a\colon\ a \in \Sigma^* \cup \{\epsilon\}\} \cup \{o_{init}, o_{set}, o_{dec}\}$. Operators in the first class implement the transition function δ while operators in the second class implement arithmetic operations on counters. Operators in the first class execute in normal mode and all have the format

$$o_a = \left\langle normal, \bigwedge_{q \in Q} (q \rhd \alpha(q,a)) \right\rangle,$$

where $\alpha(q,a)$ is the effect associated to state q on input $a \in \Sigma \cup \{\epsilon\}$. For states $q \in Q \setminus cstates$, these are

$$\alpha(q,a) \stackrel{\text{def}}{=} \begin{cases} \top \oplus \bigoplus_{q' \in \delta(q,a)\setminus\{q\}} (\neg q \wedge q') & \text{if } a \neq \epsilon,\ q \in \delta(q,a), \\ \bigoplus_{q' \in \delta(q,a)} (\neg q \wedge q') & \text{if } a \neq \epsilon,\ q \notin \delta(q,a), \\ \top \oplus \bigoplus_{q' \in \delta(q,\epsilon)} (\neg q \wedge q') & \text{if } a = \epsilon, \end{cases}$$

where $\top$ is the empty or null effect, used in cases when the transition may leave the state of the automata unchanged.

As seen in the previous section, some arithmetic operations are implemented as sequences of operators. We enforce such sequences with a change on the execution mode obtained by deleting the fluent *normal*. Arithmetic operations can only occur for transitions on the cstates associated with the counters. In states $test_c$, there are ϵ-transitions to the states $continue_c$ and $exit_c$ depending on whether the counter is greater than or equal to zero. In states $continue_c$ and $exit_c$, there are ϵ-transitions to the unique states in $\delta(continue_c, \epsilon)$ and $\delta(exit_c, \epsilon)$ respectively. And, in states $loop_c$, there is an ϵ-transition to the state $test_c$ and the counter is decremented. Furthermore, all transitions in cstates on a symbol $a \in \Sigma$ lead to the sink node. These effects are the following:

$$\alpha(test_c, \epsilon) \stackrel{\text{def}}{=} \top \oplus \left[\neg test_c \wedge (c > 0 \rhd continue_c) \wedge (c = 0 \rhd exit_c)\right],$$

$$\alpha(continue_c, \epsilon) \stackrel{\text{def}}{=} \top \oplus \left[\neg continue_c \wedge \delta(continue_c, \epsilon)\right],$$

$$\alpha(loop_c, \epsilon) \stackrel{\text{def}}{=} \top \oplus [\neg loop_c \wedge test_c \wedge \neg normal \wedge dec_{c,1}],$$

$$\alpha(exit_c, \epsilon) \stackrel{\text{def}}{=} \top \oplus \left[\neg exit_c \wedge \delta(exit_c, \epsilon)\right],$$

$$\alpha(q,a) \stackrel{\text{def}}{=} \neg q \wedge q_{sink}, \quad \text{for } q \in cstates,\ a \in \Sigma.$$

The effect for $loop_c$ changes the execution mode from *normal* to $dec_{c,1}$ in order to decrement the counter c by one unit through a sequence of operators. The mode $dec_{c,1}$ is an instance of the more general execution mode $dec_{c,k}$ which performs a decrement of the counter c by k units. The operators and effects that implement the decrement operations are:

$$o_{dec} \stackrel{\text{def}}{=} \left\langle true, \bigwedge_{c,k} dec_{c,k} \rhd \beta_{dec}(c,k) \right\rangle,$$

$$\beta_{dec}(c,k) \stackrel{\text{def}}{=} \neg dec_{c,k} \wedge rdec_{c,k-1} \wedge flag_c,$$

$$\beta_{dec}(c,0) \stackrel{\text{def}}{=} \neg dec_{c,0} \wedge normal,$$

$$o_{flag} \stackrel{\text{def}}{=} \left\langle true, \bigwedge_{c} flag_c \rhd (\neg flag_c \wedge filter_{c,p(n)} \wedge flag_c) \right\rangle,$$

$$o_{filter} \stackrel{\text{def}}{=} \left\langle true, \bigwedge_{c,k} filter_{c,k} \rhd \beta_{filter}(c,k) \right\rangle,$$

$$\beta_{filter}(c,k) \stackrel{\text{def}}{=} \neg filter_{c,k} \wedge filter_{c,k-1} \wedge filter_{c,k},$$

$$\beta_{filter}(c,0) \stackrel{\text{def}}{=} \neg filter_{c,0} \wedge replace_c \wedge filter_{c,0},$$

$$o_{replace} \stackrel{\text{def}}{=} \left\langle true, \bigwedge_{c,k} (replace_c \wedge rdec_{c,k}) \rhd (\neg replace_c \wedge dec_{c,k} \wedge replace_c) \right\rangle.$$

We claim that once the fluent $dec_{c,k}$ becomes true in a belief state b, the only way to resume normal execution mode is to perform an appropriate sequence of o_{dec}, o_{flag}, o_{filter} and $o_{replace}$ operators so that the value of the counter c denoted by b is decremented by k units. Indeed, once $dec_{c,k}$ is true and *normal* is false, o_{dec} is the only operator that has an effect. Its

Please cite this article in press as: B. Bonet, Conformant plans and beyond: Principles and complexity, Artificial Intelligence (2009), doi:10.1016/j.artint.2009.11.001

effect is to clear $dec_{c,k}$ and set the fluents $rdec_{c,k-1}$ and $flag_c$; the first will force a new application of o_{dec} while the second initiates the decrement operation on the counter c. Once $flag_c$ is true, the only operator that has an effect is o_{flag} which performs the flag effect (cf., Section 8), clears $flag_c$ and sets $filter_{c,p(n)}$. At this stage, the operator o_{filter} must be applied $p(n)+1$ times thus performing the sequence of effects $filter_{c,p(n)}, filter_{c,p(n)-1}, \ldots, filter_{c,0}$; the $(n+1)$st application clears $filter_{c,0}$ and sets $replace_c$ to force the application of $o_{replace}$. The operator $o_{replace}$ applies the effect $replace_c$ and changes the fluent $rdec_{c,k}$ by $dec_{c,k}$ in order to force additional decrement operations. At the end, the last operator o_{dec} changes the fluent $dec_{c,0}$ by *normal* and the normal execution mode resumes. Observe that when *normal* becomes true, it becomes true at all belief states and that the state of the automata is the one when the *normal* fluent was deleted; e.g. $test_c$ in the case of $\alpha(loop_c, \epsilon)$.

There is just one last thing to do which is define the ϵ-transitions for the states $entry_c$ that initialize the value of the counters to $c = bound_c$ and change state to $test_c$. In general we cannot set a counter of double-exponential capacity to an arbitrary value using operators of polynomial size. However, the exponents to consider are among $\{1, \ldots, n+1, 2^{2^{p(n)}} - n - 2, 2^{2^{p(n)}} - 2, 2^{2^{p(n)}}\}$ and such values can be set with operators of polynomial size. Indeed, let us begin with the effect

$$\alpha(entry_c, \epsilon) \stackrel{\text{def}}{=} \top \oplus \left[\neg entry_c \wedge test_c \wedge \neg normal \wedge init_c \wedge A_c \wedge \bigwedge_k \neg m_{c,k}\right]$$

that sets the active fluent and clears all markers for c, changes the execution mode from normal to $init_c$, and sets the state of the automata to be $test_c$ once normal execution mode is restored. As before, the initialization for some values may involve several operators and this is the reason for the change of execution mode. Once in $init_c$ mode, the initialization is done with the operator

$$o_{init} \stackrel{\text{def}}{=} \left\langle true, \bigwedge_c init_c \rhd \beta_{init}(c)\right\rangle.$$

The effect $\beta_{init}(c)$ depends on the counter c. Let us partition the counters into those with at most exponential capacity, denoted by C_e, and those with double-exponential capacity denoted by C_d. A counter $c \in C_e$ has at most a polynomial number of 1-bits, thus it can be directly initialized with the effect

$$\beta_{init}(c) \stackrel{\text{def}}{=} \neg init_c \wedge normal \wedge \bigoplus\left\{\bigwedge\{m_{c,k}\colon \left(k\text{th bit of } [i]_{\text{bit}}\right) = 1\}\colon \left(i\text{th bit of } [bound_c]_{\text{bit}}\right) = 1\right\}$$

that creates one state for each 1-bit in the target value for the counter c, and sets the markers appropriately. For counters in $c \in C_d$, the value is first set to $2^{2^{p(n)}}$ and then decremented $2^{2^{p(n)}} - bound_c$ times (which is a polynomial number of times) to reach the target value $bound_c$. The value $2^{2^{p(n)}}$ corresponds to the unique bit $m_{c,p(n)}$ and the decrements are forced by setting the execution mode to the appropriate $dec_{c,k}$:

$$\beta_{init}(c) \stackrel{\text{def}}{=} \neg init_c \wedge dec_{c,2^{2^{p(n)}} - bound_c} \wedge m_{c,p(n)}.$$

For example, to set the counter c to the value $2^{2^{p(n)}} - n - 2$, the counter is first set to the value $2^{2^{p(n)}}$ and then decremented $n+2$ times. Double-exponential counters have initial values among $\{2^{2^{p(n)}} - n - 2, 2^{2^{p(n)}} - 2, 2^{2^{p(n)}}\}$ and thus the number of effects of type $\beta_{dec}(c,k)$ and $\beta_{filter}(c,k)$ is polynomial.

In this case, a configuration $\langle q, \nu\rangle$ for M_α is represented by a belief state denoted as $\{q, \nu\}$. The construction of P is faithful in the sense that for any pair of configurations $\langle q, \nu\rangle$ and $\langle q', \nu'\rangle$ for M_α,

$$\left\{q', \nu'\right\} \in Res\left(\pi_{pad} o_{a_1} \pi_{pad} o_{a_2} \pi_{pad} \cdots \pi_{pad} o_{a_n} \pi_{pad}, \left\{\{q, \nu\}\right\}\right) \quad \text{iff } \left\langle q', \nu'\right\rangle \in \hat{\delta}\left(\langle q, \nu\rangle, a_1 \cdots a_n\right),$$

where $\hat{\delta}$ is the extended transition function for the NFAC M_α (see Appendix B). The subsequence π_{pad} is a long enough sequence of operators in $\{o_\epsilon, o_{dec}, o_{flag}, o_{filter}, o_{replace}, o_{init}\}$ that guarantees the belief states are always updated with respect to the NFAC. Indeed, it suffices to define π_{pad} as 2^N, $N = |Q| \times \Pi_{c \in C}(1 + bound_c)$, repetitions of $\langle o_\epsilon, o_{init}, o_{flag}, \pi_{filter}, o_{replace}\rangle$ where π_{filter} is a sequence of $1 + p(n)$ repetitions of o_{filter}. Therefore, P has a valid linear plan iff $L(M_\alpha) \neq \Sigma^*$ iff M rejects ω. Since 2EXPSPACE is closed under complementation, we have that PLAN-PO-LINEAR is 2EXPSPACE-hard. □

Finally, combining the simulation of counters of double-exponential capacity with reductions among partially-observable problems, as done in Theorem 8, we obtain

Theorem 10. *For every $k \geqslant 0$,* PLAN-PO-BRANCH(k) *is 2EXPSPACE-hard. Hence,* PLAN-PO-BRANCH(k) *and* PLAN-PO-BRANCH *are 2EXPSPACE-complete.*

10. Problems without modal formulae and polynomial plans

Turner [61] studies the complexity of deciding the existence of plans of polynomial length using quantified boolean formulae (QBFs). He shows that deciding the existence of linear plans of polynomial length for fully-observable problems

Please cite this article in press as: B. Bonet, Conformant plans and beyond: Principles and complexity, Artificial Intelligence (2009), doi:10.1016/j.artint.2009.11.001

is in Σ_3^P, and that deciding the existence of contingent plans of polynomial length is PSPACE-complete for problems with either full or partial observability. However, Turner uses a different representation language which is based on factored relations between states. With our representation language, in which the applicability of the actions can be decided efficiently, Turner results translate into a Σ_2^P completeness for checking the existence of linear plans for fully-observable problems.

Following the ideas of Turner, we study the complexity of checking the existence of plans of polynomial length for problems with no modal formulae and the complexity of checking the existence of linear plans, of arbitrary length, for problems with no modal formulae. Partially-observable problems with no modal formulae are currently predominant in automated planning and thus important to consider as a special case.

10.1. Plans of polynomial length for problems without modal formulae

Let us begin with a formal definition of the decision problems considered. Let $q(n)$ be a polynomial. A class $\mathcal{P}$ of planning problems has valid plans of polynomial length (modulo q) iff each problem P in $\mathcal{P}$ has a valid plan of length at most $q(|P|)$; recall that the length of a branching plan is its height. Since q is fixed, we should consider decision problems with respect to q. However, we can remove this dependency by defining the following decision problems:

- PLAN-FO-BRANCH-LEN$_{PL}(k) \stackrel{\text{def}}{=} \{\langle P, 1^N\rangle$: P is a FOP[PL] with a k-plan of length at most $N\}$,
- PLAN-FO-BRANCH-LEN$_{PL} \stackrel{\text{def}}{=} \{\langle P, 1^N, 1^k\rangle$: P is a FOP[PL] with a k-plan of length at most $N\}$,
- PLAN-PO-BRANCH-LEN$_{PL}(k) \stackrel{\text{def}}{=} \{\langle P, 1^N\rangle$: P is a POP[PL] with a k-plan of length at most $N\}$,
- PLAN-PO-BRANCH-LEN$_{PL} \stackrel{\text{def}}{=} \{\langle P, 1^N, 1^k\rangle$: P is a POP[PL] with a k-plan of length at most $N\}$,

where FOP[PL] and POP[PL] denote the class of planning problem with full observability and no modal formulae and with partial observability and no modal formulae respectively. Observe that the integers N and k are *written in unary*.

Thus, for example, if we need to check whether a problem P in FOP[PL] has a valid k-plan of length $q(|P|)$, then it is enough to check whether $\langle P, 1^{q(|P|)}\rangle \in$ PLAN-FO-BRANCH-LEN$_{PL}(k)$. Therefore, a function that on input $\langle P\rangle$ outputs $\langle P, 1^{q(|P|)}\rangle$ is a polynomial-time reduction from the problem of deciding the existence of a k-plan of polynomial length (modulo q) to PLAN-FO-BRANCH-LEN$_{PL}(k)$.

We show that PLAN-FO-BRANCH-LEN$_{PL}(k)$ and PLAN-PO-BRANCH-LEN$_{PL}(k)$ are both Σ_{2k+2}^P-complete, and that PLAN-FO-BRANCH-LEN$_{PL}$ and PLAN-PO-BRANCH-LEN$_{PL}$ are both PSPACE-complete.

Consider a fully-observable problem $P = \langle D, I, G, O\rangle$ with no modal formulae, a fixed planning horizon N and integer k. As it is usual in SAT-based approaches for planning [36], we encode problem P into a propositional theory whose propositions refer to the operators and fluents in P tagged with time indices. We use propositions f_t to denote the truth value of fluent f at time t, and propositions o_t to denote the application of operators at time t. There are more efficient translations that use $\lceil \log |O| \rceil$ propositions to represent the operators, yet we will not dive into such tedious details.

We use the formula $pre_o(f_1)$, with free variables among the fluents f tagged at time 1, to denote the precondition of the operator o, and the formula $dyn_o(f_1, f_2)$, with free variables among the fluents tagged at times 1 and 2, to denote the effects of the operator o. For example, if $o = \langle p, q\rangle$ then $pre_o = p_1$ and $dyn_o = \bigwedge_{f \neq q}(f_2 \equiv f_1) \wedge q_2$. These low-level formulae are collected into the following formulae

$$\Psi_{pre}(o_1, f_1) = \bigwedge_{o \in O} \big(o_1 = o \Rightarrow prec_o(f_1)\big),$$

$$\Psi_{dyn}(o_1, f_1, f_2) = \bigwedge_{o \in O} \big(o_1 = o \Rightarrow dyn_o(f_1, f_2)\big)$$

that have free variables among operators at time 1 and fluents at time 1 and 2. The notation $\Psi_{pre}(o_t, f_t)$ and $\Psi_{dyn}(o_t, f_t, f_{t+1})$ refers to the substitution of propositions o_1, f_1 and f_2 by o_t, f_t and f_{t+1} respectively. Finally, let I_t and G_t refer to the formulae for the initial and goal situations tagged at time t.

We now describe how the decision problem for P is encoded using QBFs. As an example, let us begin with a formula that tells whether there is a linear plan of length 1 for P. That is, one action that when applied at every initial state generates a goal state. This formula is

$$(\exists o_1 \forall f_1 f_2)\big[\big(I_1 \Rightarrow \Psi_{pre}(o_1, f_1)\big) \wedge \big(I_1 \wedge \Psi_{dyn}(o_1, f_1, f_2) \Rightarrow G_2\big)\big]$$

since $I_1 \Rightarrow \Psi_{pre}(o_1, f_1)$ is valid only when o_1 is applicable at each initial state, and $I_1 \wedge \Psi_{dyn}(o_1, f_1, f_2) \Rightarrow G_2$ is valid only when each state that result from the application of o_1 is a goal state.

Likewise, the following formula tells whether there is a plan that makes one observation whose first segment have two operators and the second one operator:

$$(\exists o_1 o_2 \forall f_1 f_2 f_3 \exists o_3 \forall f_4)\big[\big(I_1 \Rightarrow \Psi_{pre}(o_1, f_1)\big) \wedge \big(I_1 \wedge \Psi_{dyn}(o_1, f_1, f_2) \Rightarrow \Psi_{pre}(o_2, f_2)\big)$$

$$\wedge \big(I_1 \wedge \Psi_{dyn}(o_1, f_1, f_2) \wedge \Psi_{dyn}(o_2, f_2, f_3) \Rightarrow \Psi_{pre}(o_3, f_3)\big)$$
$$\wedge \big(I_1 \wedge \Psi_{dyn}(o_1, f_1, f_2) \wedge \Psi_{dyn}(o_2, f_2, f_3) \wedge \Psi_{dyn}(o_3, f_3, f_4) \Rightarrow G_4\big)\big].$$

If we assume the existence of no-ops, this formula also determines the existence of a 1-plan with a first segment of length at most 2 and a second segment of length at most 1, and also determines the existence of a linear plan of length at most 2.

If we want to check for the existence of a linear plan of length at most N, we use

$$(\exists o_1 \cdots o_N \forall f_1 \cdots f_{N+1})\big[\big(I_1 \Rightarrow \Psi_{pre}(o_1, f_1)\big)$$
$$\wedge \big(I_1 \wedge \Psi_{dyn}(o_1, f_1, f_2) \Rightarrow \Psi_{pre}(o_2, f_2)\big)$$
$$\cdots$$
$$\wedge \big(I_1 \wedge \Psi_{dyn}(o_1, f_1, f_2) \wedge \cdots \wedge \Psi_{dyn}(o_{N-1}, f_{N-1}, f_N) \Rightarrow \Psi_{pre}(o_N, f_N)\big)$$
$$\wedge \big(I_1 \wedge \Psi_{dyn}(o_1, f_1, f_2) \wedge \cdots \wedge \Psi_{dyn}(o_N, f_N, f_{N+1}) \Rightarrow G_{N+1}\big)\big].$$

In general, when we want to check for the existence of a k-plan of length at most N, we use

$$PlanFO(k, N) = (\exists o_1 \cdots o_N \forall f_1 \cdots f_{N+1} \cdots \exists o_{kN+1} \cdots o_{(k+1)N} \forall f_{kN+2} \cdots f_{(k+1)N+1})[\Psi \wedge \Phi],$$

where the formula $\Psi(o_1, \ldots, o_{(k+1)N}, f_1, \ldots, f_{(k+1)N+1})$ is a conjunction of implications as in the examples, and $\Phi(o_1, \ldots, o_{(k+1)N})$ *verifies that the number of non-no-op operators is less than or equal to* N by using additional propositions in order to have polynomial size in k and N.

Theorem 11. *For* $k \geqslant 0$, PLAN-FO-BRANCH-LEN$_{PL}(k)$ *is* Σ^P_{2k+2}*-complete.*

Proof. For inclusion, note that $PlanFO(k, N)$ has $2k + 2$ alternations. Hence, given an instance $\langle P, 1^N \rangle$ in PLAN-FO-BRANCH-LEN$_{PL}(k)$, where k is constant, construct in polynomial time the formula $PlanFO(k, N)$ and then call a decision algorithm for Σ^P_{2k+2} problems.

It remains to show hardness. We show for $k = 1$ since the proof for general k is similar. Consider a formula $\Phi = (\exists \bar{x}_1 \forall \bar{y}_1 \exists \bar{x}_2 \forall \bar{y}_2)\Psi$. We need to construct a fully-observable problem P, with size polynomial in $|\Phi|$, such that P has a valid 1-plan iff Φ is valid. Since Φ has 4 quantifiers, this implies hardness for Σ^P_4.

Without loss of generality, assume that $\overline{x_i}$ and $\overline{y_i}$ have two boolean variables each, say x_i^1, x_i^2 and y_i^1, y_i^2, so that $\Phi = (\exists x_1^1 x_1^2 \forall y_1^1 y_1^2 \exists x_2^1 x_2^2 \forall y_2^1 y_2^2)\Psi$. The problem P has two propositions x_i^j and s_i^j for each x_i^j-variable that denote their truth value and whether it has been set or not, and one proposition y_i^j for each y_i^j-variable that denotes their truth value. The initial and goal states for P are defined by $I \stackrel{\text{def}}{=} \bigwedge_{i,j} \neg x_i^j \wedge \neg s_i^j \wedge \neg y_i^j$, and $G \stackrel{\text{def}}{=} \bigwedge_{i,j} s_i^j \wedge \Psi$ respectively. The operators are:

$$set(x_1^1, T) \stackrel{\text{def}}{=} \langle \neg s_1^1, s_1^1 \wedge x_1^1 \rangle, \qquad set(x_1^1, F) \stackrel{\text{def}}{=} \langle \neg s_1^1, s_1^1 \wedge \neg x_1^1 \rangle,$$
$$set(x_1^2, T) \stackrel{\text{def}}{=} \left\langle \neg s_1^2 \wedge s_1^1, s_1^2 \wedge x_1^2 \wedge \bigwedge_{j=1}^{2} (y_1^j \oplus \neg y_1^j) \right\rangle, \qquad set(x_1^2, F) \stackrel{\text{def}}{=} \left\langle \neg s_1^2 \wedge s_1^1, s_1^2 \wedge \neg x_1^2 \wedge \bigwedge_{j=1}^{2} (y_1^j \oplus \neg y_1^j) \right\rangle,$$
$$set(x_2^1, T) \stackrel{\text{def}}{=} \langle \neg s_2^1 \wedge s_1^2, s_2^1 \wedge x_2^1 \rangle, \qquad set(x_2^1, F) \stackrel{\text{def}}{=} \langle \neg s_2^1 \wedge s_1^2, s_2^1 \wedge \neg x_2^1 \rangle,$$
$$set(x_2^2, T) \stackrel{\text{def}}{=} \left\langle \neg s_2^2 \wedge s_2^1, s_2^2 \wedge x_2^2 \wedge \bigwedge_{j=1}^{2} (y_2^j \oplus \neg y_2^j) \right\rangle, \qquad set(x_2^2, F) \stackrel{\text{def}}{=} \left\langle \neg s_2^2 \wedge s_2^1, s_2^2 \wedge \neg x_2^2 \wedge \bigwedge_{j=1}^{2} (y_2^j \oplus \neg y_2^j) \right\rangle.$$

We claim that Φ is valid iff P has a valid 1-plan. The necessity is direct. For the sufficiency, assume that P has a valid 1-plan. By construction, the operators need to be applied in order to set the values for x_1^1, x_1^2, x_2^1 and x_2^2. Let us consider the 4 possible positions for the branch point (observation):

Case 1: before x_1^1 is set. Since this is the only branch point, the value for the x's are set without observing the values for the y's. Since the plan is valid, this means that $\Phi' = (\exists x_1^1 x_1^2 x_2^1 x_2^2 \forall y_1^1 y_1^2 y_2^1 y_2^2)\Psi$ is valid. Clearly, $\Phi' \Rightarrow \Phi$.

Case 2: after x_1^1 is set but before x_1^2. This case is similar to the previous one, and corresponds to the same Φ' formula as the operators $set(x_1^1, \cdot)$ do not set values for y's.

Case 3: after x_1^2 is set but before x_2^1. Since the operators for x_1^2 non-deterministically set the value for the y_1's, and the value for x_2^j can depend on the value for previous variables, we have that Ψ is valid.

Case 4: after x_2^1 is set but before x_2^2. This means that the value for x_2^1 does not depend on the values for the y_1's, yet the value of x_2^2 might depend on these values. Since the plan is valid, this means that $\Phi' = (\exists x_1^1 x_1^2 x_2^1 \forall y_1^1 y_1^2 \exists x_2^2 \forall y_2^1 y_2^2)\Psi$ is valid. Clearly, $\Phi' \Rightarrow \Phi$.

In all cases, Φ is valid, and the theorem holds. □

Please cite this article in press as: B. Bonet, Conformant plans and beyond: Principles and complexity, Artificial Intelligence (2009), doi:10.1016/j.artint.2009.11.001

Theorem 12. PLAN-FO-BRANCH-LEN$_{PL}$ *is PSPACE-complete.*

Proof. The problem of deciding the validity of QBFs is PSPACE-complete. Given an input $\langle P, 1^N, 1^k\rangle$ construct the formula $PlanFO(k, N)$ in polynomial time, since N and k are written in unary, and feed it to a QBF solver: P has a k-plan of length at most N iff the formula is valid. Therefore, PLAN-FO-BRANCH-LEN$_{PL}$ is reduced to QBF and we have the inclusion.

For hardness, let Φ be a QBF starting with an existential quantifier with $2k+2$ alternations and N existential variables. Construct a problem P as in the proof of Theorem 11. Then, Φ is valid iff $\langle P, 1^N, 1^k\rangle \in$ PLAN-FO-BRANCH-LEN$_{PL}$. If Φ does not start with an existential quantifier, or the number of alternations is not even and at least 2, then add dummy variables and quantifiers. □

Let $P = \langle D, I, G, O, Z\rangle$ be a planning problem with partial observability and no modal formulae. The fluent symbols are partitioned into observables Z and unobservables $D \setminus Z$. Let us denote with f_t^Z and f_t^U the observable and unobservable fluents at time t. The existence of a linear plan of length at most N can be decided with a QBF of the form:

$$\big(\exists o_1 \cdots o_N \forall f_1^Z \cdots f_{N+1}^Z \forall f_1^U \cdots f_{N+1}^U\big)[\Psi],$$

where Ψ tells whether the sequence $o_1 \cdots o_N$ is a valid plan. Likewise, the existence of a 1-plan of length at most N can be decided with:

$$\big(\exists o_1 \cdots o_N \forall f_1^Z \cdots f_{N+1}^Z \exists o_{N+1} \cdots o_{2N} \forall f_{N+1}^Z \cdots f_{2N+1}^Z \forall f_1^U \cdots f_{2N+1}^U\big)[\Psi \wedge \Phi],$$

where Ψ is as before and $\Phi(o_1, \ldots, o_{2N})$ verifies that the number of non-no-op operators is less than or equal to N. In general, the existence of a k-plan of length N can be decided with the QBF

$$PlanPO(k, N) = \big(\exists o_1 \cdots o_N \forall f_1^Z \cdots f_{N+1}^Z \cdots \exists o_{kN+1} \cdots o_{(k+1)N} \forall f_{kN+2}^Z \cdots f_{(k+1)N+1}^U\big)[\Psi \wedge \Phi].$$

As for the case of fully-observable problems, it is not difficult to show the following results.

Theorem 13. *For $k \geqslant 0$,* PLAN-PO-BRANCH-LEN$_{PL}(k)$ *is Σ^P_{2k+2}-complete.* PLAN-PO-BRANCH-LEN$_{PL}$ *is PSPACE-complete.*

Proof. The inclusion is shown as before. For hardness, note that fully-observable problems are special instances of partially-observable problems. □

10.2. Plans for problems without modal formulae

Let B be a subset of belief states, as defined in Section 3, and o an operator. By definition, o is applicable at B if it is applicable in all $b \in B$. Therefore, since o contains no modalities, o is applicable in B iff it is applicable in all states in $\bigcup_{b\in B} b$. Similarly, B is a goal set if all $b \in B$ are goal beliefs, and thus, since there are no modalities, B is a goal set iff $\bigcup_{b\in B} b$ is a goal belief. In summary, the existence of a linear plan for problems with no modalities can be established by considering belief states instead of subsets of belief states, and similarly for plans of bounded branching.[2]

Theorem 14. *Deciding the existence of a linear plan, without restrictions on its length, for planning problems with no modalities is EXPSPACE-complete. Similarly, deciding the existence of a plan of bounded branching, without restrictions on its length, for planning problem with no modalities is EXPSPACE-complete.*

We think that this result is the main reason why linear plans had been only considered for problems with no observability. When there are no modal formulae involved, the requirements of conformance for the fully-observable and the partially-observable cases collapse. However, when modal formulae are allowed, both requirements become provable different since, by the space hierarchy theorems [56], EXPSPACE is different from 2EXPSPACE.

11. Discussion

The term 'conformant' had been used to refer to unobservable planning problems and their linear solutions. In this work, we have shown that linear plans are also meaningful for partially-observable problems, and thus conformance should be thought as a property on the plans, namely linearity, and not as a property of the models.

Conformant and contingent plans are the extreme points of a discrete spectrum of solution forms that also contains plans of bounded branching. We have derived exact complexity results for checking the existence of linear plans and plans of bounded branching for fully-observable and partially-observable domains. We also considered special classes of problems

[2] Interestingly, De Giacomo and Vardi [22] give an EXPSPACE-completeness result for deciding the existence of linear plans for partially-observable problems in which the actions are deterministic and the goal is specified with a Büchi automaton on observation traces. However, this class of problems does not explicitly accomodate modal operators of the type studied in this work.

Please cite this article in press as: B. Bonet, Conformant plans and beyond: Principles and complexity, Artificial Intelligence (2009), doi:10.1016/j.artint.2009.11.001

such as plans for problems without modal formulae, and plans of polynomial length. Interestingly, when the problems have no modal formulae, the complexity of checking the existence of plans for fully-observable and partially-observable problems coincide.

An issue that remains open is the role of cyclic plans and whether it makes sense to talk about bounded branching in such cases. At first sight, we could say that a simple plan that cycles over a sequence of actions is a linear plan. However, even understanding cyclic plans for non-deterministic tasks is sometimes difficult [55], and thus we are not ready yet to talk about cyclic linear plans or cyclic plans of bounded branching.

Another interesting technical question that is not answered is to determine the complexity of deciding the existence of polynomial plans for problems with modal formulae. In this case, it does not seem that a direct reduction from such a problem to the problem of deciding the validity of a (bounded) QBF would exist. The reason is that the validity of modal formulae must be tested multiple times in the former problem where each test is NP-hard.

One interesting point raised by the reviewers is whether there is a real need to consider modal operators in a planning language. Currently, there is no an established consensus on this issue. Some researches think that the current languages without modal operators are enough, others that the languages must be extended with explicit notions of knowledge. One of the strongest arguments against the inclusion of modal operators is that current planning systems are able to model certain notions of knowledge if "things" are "encoded" in a proper manner. Although this is partially true, I think there are plenty of examples on which the current languages are simply not expressive enough or for which a translation to a modality-free language would be of exponential size. This is an interesting discussion that I hope it would be clarified in the years to come as our understanding of knowledge and its dynamics increases.

On the practical side, we have not dealt with the problem of computing plans efficiently. Meuleau and Smith [45] propose an algorithm that works for computing plans of bounded branching for fully-observable problems but not for computing such plans for partially-observable problems. We believe that algorithms based on heuristic search and the use of methods to compactly represent sets of states such as BDDs might work for partially-observable problems.

Acknowledgements

Thanks to the anonymous reviewers and the editor for interesting and challenging comments that resulted in an improved paper. Also, thanks to Patrik Haslum and Héctor Geffner for interesting discussions on early drafts.

Appendix A. Borodin's Theorem

The following theorem is attributed to A. Borodin in the seminal work of Chandra, Kozen and Stockmeyer [13]. It is a generalization for ATMs of Savitch's Theorem.

Theorem 15. *If M is an $s(n)$-space bounded and $a(n)$-alternation bounded ATM with $s(n) \geqslant \log n$, then $M \in DSPACE(a(n)s(n) + s(n)^2)$.*

Appendix B. Extension of the transition function for NFACs

Given an NFAC $M = \langle Q, \Sigma, \delta, q_0, F, C\rangle$, we show how to construct the extended transition function $\hat{\delta}$ that maps configurations of M and words into subsets of configurations. Recall that a snapshot of M is a tuple $\langle q, \nu\rangle$ where q is a state and ν a function that maps counters to values. First, let us extend δ into a function that receives pairs $\langle \theta, a\rangle$ where θ is a snapshot and $a \in \Sigma \cup \{\epsilon\}$ as follows:

$$\delta\big(\langle q, \nu\rangle, a\big) \stackrel{\text{def}}{=} \begin{cases} \{\langle q', \nu\rangle\colon\ q' \in \delta(q, a)\} & \text{if } q \notin \{entry_c, loop_c\} \text{ or } a \neq \epsilon, \\ \{\langle test_c, \nu[c = bound_c]\rangle\} & \text{if } q = entry_c \text{ and } a = \epsilon, \text{ and} \\ \{\langle test_c, \nu[c = \nu(c) - 1]\rangle\} & \text{if } q = loop_c \text{ and } a = \epsilon, \end{cases}$$

where the function $\nu[c = v]$ is like ν except that $\nu[c = v](c) = v$. Note that the numbers of configurations is bounded from above by $|Q| \times \Pi_{c \in C}(1 + bound_c)$.

This transition function is further extended into a function $\hat{\delta}$ defined from snapshots and words in Σ^* into subsets of snapshots. Our goal is that $\hat{\delta}(\langle q, \nu\rangle, \omega)$ is the collection of snapshots that can be reached from $\langle q, \nu\rangle$ along a path labeled ω, perhaps including edges labeled ϵ. Thus, it will be important to compute the subset of snapshots reachable from a given one using ϵ transitions only. This subset, denoted by ϵ-CLOSURE($\langle q, \nu\rangle$), is equivalent to the collection of snapshots that can be reached from $\langle q, \nu\rangle$ through ϵ-paths; an ϵ-path is a path made only of ϵ transitions. Furthermore, we naturally let ϵ-CLOSURE(S), where S is a set of configurations, be $\bigcup_{\theta \in S} \epsilon$-CLOSURE($\theta$). The function $\hat{\delta}$ is defined as:

1. $\hat{\delta}(\langle q, \nu\rangle, \epsilon) \stackrel{\text{def}}{=} \epsilon$-CLOSURE($\langle q, \nu\rangle$).
2. For $\omega \in \Sigma^*$ and $a \in \Sigma$, $\hat{\delta}(\theta, \omega a) \stackrel{\text{def}}{=} \epsilon$-CLOSURE($S$) where $S = \{\theta'\colon$ for some $\theta'' \in \hat{\delta}(\theta, \omega),\ \theta' \in \delta(\theta'', a)\}$.

If $\theta_0 = \langle q_0, 0\rangle$ is the initial configuration, the language $L(M)$ is $\{\omega \in \Sigma^*\colon$ for some $\langle q, \nu\rangle \in \hat{\delta}(\theta_0, \omega),\ q \in F\}$.

Please cite this article in press as: B. Bonet, Conformant plans and beyond: Principles and complexity, Artificial Intelligence (2009), doi:10.1016/j.artint.2009.11.001

References

[1] D. Bernstein, E. Hansen, S. Zilberstein, Bounded policy iteration for decentralized POMDPs, in: L.P. Kaelbling, A. Saffiottiiba (Eds.), Proc. 19th Int. Joint Conf. on Artificial Intelligence, Edinburgh, Scotland, Professional Book Center, 2005, pp. 1287–1292.
[2] P. Bertoli, A. Cimatti, M. Roveri, P. Traverso, Strong planning under partial observability, Artificial Intelligence 170 (4–5) (2006) 337–384.
[3] D. Bertsekas, Dynamic Programming and Optimal Control (2 vols), Athena Scientific, 1995.
[4] P. Blackburn, M. de Rijke, Y. Venema, Modal Logic, Cambridge University Press, 2002.
[5] B. Bonet, Bounded branching and modalities in non-deterministic planning, in: D. Long, S. Smith, D. Borrajo, L. McCluskey (Eds.), Proc. 16th Int. Conf. on Automated Planning and Scheduling, Ambleside, UK, AAAI Press, 2006, pp. 42–51.
[6] B. Bonet, H. Geffner, Planning with incomplete information as heuristic search in belief space, in: S. Chien, S. Kambhampati, C. Knoblock (Eds.), Proc. 6th Int. Conf. on Artificial Intelligence Planning and Scheduling, Breckenridge, CO, AAAI Press, 2000, pp. 52–61.
[7] B. Bonet, H. Geffner, Planning as heuristic search, Artificial Intelligence 129 (1–2) (2001) 5–33.
[8] B. Bonet, H. Geffner, Faster heuristic search algorithms for planning with uncertainty and full feedback, in: G. Gottlob (Ed.), Proc. 18th Int. Joint Conf. on Artificial Intelligence, Acapulco, Mexico, Morgan Kaufmann, 2003, pp. 1233–1238.
[9] B. Bonet, H. Geffner, An algorithm better than AO*?, in: M. Veloso, S. Kambhampati (Eds.), Proc. 20th National Conf. on Artificial Intelligence, Pittsburgh, PA, AAAI Press/MIT Press, 2005, pp. 1343–1348.
[10] D. Bryce, S. Kambhampati, D.E. Smith, Planning graph heuristics for belief space search, J. Artificial Intelligence Res. 26 (2006) 35–99.
[11] T. Bylander, The computational complexity of propositional STRIPS planning, Artificial Intelligence 69 (1994) 165–204.
[12] A.R. Cassandra, M. Littman, L.P. Kaelbling, Acting optimally in partially observable stochastic domains, in: B. Hayes-Roth, R. Korf (Eds.), Proc. 12th National Conf. on Artificial Intelligence, Seattle, WA, AAAI Press/MIT Press, 1994, pp. 1023–1028.
[13] A. Chandra, D. Kozen, L.J. Stockmeyer, Alternation, J. ACM 28 (1) (1981) 114–133.
[14] V. Chvatal, Mastermind, Combinatorica 3 (3–4) (1983) 325–329.
[15] A. Cimatti, M. Pistore, M. Roveri, P. Traverso, Weak, strong, and strong cyclic planning via symbolic model checking, Artificial Intelligence 147 (2003) 35–84.
[16] A. Cimatti, M. Roveri, P. Bertoli, Conformant planning via symbolic model checking and heuristic search, Artificial Intelligence 159 (2004) 127–206.
[17] D. Du, K. Ko, Theory of Computational Complexity, Wiley–Interscience, New York, NY, 2000.
[18] P. Erdös, C. Ranyi, On two problems in information theory, Magyar Tud. Akad. Mat. Kut. Int. Közl. 8 (1963) 229–242.
[19] R. Fagin, J. Halpern, Y. Moses, M. Vardi, Reasoning about Knowledge, MIT Press, 1995.
[20] R. Fikes, N. Nilsson, STRIPS: A new approach to the application of theorem proving to problem solving, Artificial Intelligence 1 (1971) 27–120.
[21] G. De Giacomo, Y. Lespérance, H.J. Levesque, S. Sardiña, On the semantics of deliberation in indigolog: From theory to implementation, Ann. Math. Artif. Intell. 41 (1–2) (2004) 256–299.
[22] G. De Giacomo, M.Y. Vardi, Automata-theoretic approach to planning for temporally extended goals, in: S. Biundo, M. Fox (Eds.), Proc. 5th European Conf. on Planning, Durham, UK, in: Lecture Notes in Comput. Sci., vol. 1809, Springer, 1999, pp. 226–238.
[23] W. Goddard, Static Mastermind, J. Combin. Math. Combin. Comput. 47 (2003) 225–236.
[24] R.P. Goldman, M.S. Boddy, Expressive planning and explicit knowledge, in: B. Drabble (Ed.), Proc. 3rd Int. Conf. on Artificial Intelligence Planning Systems, Edinburgh, Scotland, AAAI Press, 1996, pp. 110–117.
[25] E. Hansen, Solving POMDPs by searching in policy space, in: G. Cooper, S. Moral (Eds.), Proc. 14th Conf. on Uncertainty in Artificial Intelligence, Madison, WI, Morgan Kaufmann, 1998, pp. 211–219.
[26] E. Hansen, S. Zilberstein, LAO*: A heuristic search algorithm that finds solutions with loops, Artificial Intelligence 129 (2001) 35–62.
[27] P. Haslum, P. Jonsson, Some results on the complexity of planning with incomplete information, in: S. Biundo, M. Fox (Eds.), Proc. 5th European Conf. on Planning, Durham, UK, in: Lecture Notes in Comput. Sci., vol. 1809, Springer, 1999, pp. 308–318.
[28] M. Helmert, The fast downward planning system, J. Artificial Intelligence Res. 26 (2006) 191–246.
[29] J. Hoffmann, R. Brafman, Contingent planning via heuristic forward search with implicit belief states, in: S. Biundo, K. Myers, K. Rajan (Eds.), Proc. 15th Int. Conf. on Automated Planning and Scheduling, Monterey, CA, Morgan Kaufmann, 2005, pp. 71–80.
[30] J. Hoffmann, R. Brafman, Conformant planning via heuristic forward search: A new approach, Artificial Intelligence 170 (2006) 507–541.
[31] J. Hoffmann, B. Nebel, The FF planning system: Fast plan generation through heuristic search, J. Artificial Intelligence Res. 14 (2001) 253–302.
[32] J. Hopcroft, J. Ullman, Introduction to Automata Theory, Languages, and Computation, Addison–Wesley, 1979.
[33] J. Huang, Combining knowledge compilation and search for conformant probabilistic planning, in: D. Long, S. Smith, D. Borrajo, L. McCluskey (Eds.), Proc. 16th Int. Conf. on Automated Planning and Scheduling, Lake District, UK, AAAI Press, 2006, pp. 253–262.
[34] G.E. Hughes, M.J. Cresswell, A New Introduction to Modal Logic, Routledge, 1996.
[35] L.P. Kaelbling, M. Littman, A.R. Cassandra, Planning and acting in partially observable stochastic domains, Artificial Intelligence 101 (1999) 99–134.
[36] H. Kautz, B. Selman, Pushing the envelope: Planning, propositional logic, and stochastic search, in: W. Clanceyuiba, D. Weld (Eds.), Proc. 13th National Conf. on Artificial Intelligence, Portland, OR, AAAI Press/MIT Press, 1996, pp. 1194–1201.
[37] D.E. Knuth, The computer as a Master Mind, J. Recreational Mathematics 9 (1976–1977) 1–6.
[38] M. Koyama, T. Lai, An optimal Mastermind strategy, J. Recreational Mathematics 25 (1993) 251–256.
[39] H.J. Levesque, What is planning in the presence of sensing, in: W. Clancey, D. Weld (Eds.), Proc. 13th National Conf. on Artificial Intelligence, Portland, OR, AAAI Press/MIT Press, 1996, pp. 1139–1146.
[40] H.J. Levesque, Planning with loops, in: L.P. Kaelbling, A. Saffiotti (Eds.), Proc. 19th Int. Joint Conf. on Artificial Intelligence, Edinburgh, Scotland, Professional Book Center, 2005, pp. 509–515.
[41] M. Littman, Probabilistic propositional planning: Representations and complexity, in: B. Kuipers, B. Webber (Eds.), Proc. 14th National Conf. on Artificial Intelligence, Providence, RI, AAAI Press/MIT Press, 1997, pp. 748–754.
[42] C. Lusena, T. Li, S. Sittinger, C. Wells, J. Goldsmith, My brain is full: When more memory helps, in: K. Laskey, H. Prade (Eds.), Proc. 15th Conf. on Uncertainty in Artificial Intelligence, Stockholm, Sweden, Morgan Kaufmann, 1999, pp. 374–381.
[43] S. Majercik, M. Littman, Maxplan: A new approach to probabilistic planning, in: R. Simmons, M. Veloso, S. Smith (Eds.), Proc. 4th Int. Conf. on Artificial Intelligence Planning Systems, Pittsburgh, PA, AAAI Press, 1998, pp. 86–93.
[44] N. Meuleau, L. Peshkin, K. Kim, L.P. Kaelbling, Learning finite-state controllers for partially observable environments, in: K. Laskey, H. Prade (Eds.), Proc. 15th Conf. on Uncertainty in Artificial Intelligence, Stockholm, Sweden, Morgan Kaufmann, 1999, pp. 427–436.
[45] N. Meuleau, D. Smith, Optimal limited contingency planning, in: C. Meek, U. Kjaerulff (Eds.), Proc. 19th Conf. on Uncertainty in Artificial Intelligence, Acapulco, Mexico, Morgan Kaufmann, 2003, pp. 417–426.
[46] A.R. Meyer, L.J. Stockmeyer, The equivalence problem of regular expressions with squaring requires exponential space, in: Proc. 13th Annual IEEE Symposium on Switching and Automata Theory, 1973, pp. 125–129.
[47] H. Palacios, B. Bonet, A. Darwiche, H. Geffner, Pruning conformant plans by counting models on compiled d-DNNF representations, in: S. Biundo, K. Myers, K. Rajan (Eds.), Proc. 15th Int. Conf. on Automated Planning and Scheduling, Monterey, CA, AAAI Press, 2005, pp. 141–150.
[48] H. Palacios, H. Geffner, From conformant into classical planning: Efficient translations that may be efficient too, in: M. Boddy, M. Fox, S. Thiébaux (Eds.), Proc. 17th Int. Conf. on Automated Planning and Scheduling, Providence, RI, AAAI Press, 2007.

Please cite this article in press as: B. Bonet, Conformant plans and beyond: Principles and complexity, Artificial Intelligence (2009), doi:10.1016/j.artint.2009.11.001

[49] C.H. Papadimitriou, Computational Complexity, Addison–Wesley, 1993.
[50] E. Pednault, ADL: Exploring the middle ground between strips and the situation calculus, in: R. Brachman, H.J. Levesque, R. Reiter (Eds.), Proc. 1st Int. Conf. on Principles of Knowledge Representation and Reasoning, Toronto, Canada, Morgan Kaufmann, 1989, pp. 324–332.
[51] P. Poupart, C. Boutilier, Bounded finite state controllers, in: Advances in Neural Information Processing Systems 16 [NIPS 2003], Vancouver, Canada, MIT Press, 2003.
[52] M. Puterman, Markov Decision Processes – Discrete Stochastic Dynamic Programming, John Wiley and Sons, Inc., 1994.
[53] J. Rintanen, Complexity of planning with partial observability, in: S. Zilberstein, S. Koenig, J. Koehler (Eds.), Proc. 14th Int. Conf. on Automated Planning and Scheduling, Whistler, Canada, AAAI Press, 2004, pp. 345–354.
[54] J. Rintanen, Distance estimates for planning in the discrete belief space, in: D.L. McGuinness, G. Ferguson (Eds.), Proc. 19th National Conf. on Artificial Intelligence, San Jose, CA, AAAI Press/MIT Press, 2004, pp. 525–530.
[55] S. Sardiña, G. De Giacomo, Y. Lespérance, H.J. Levesque, On the limits of planning over belief states under strict uncertainty, in: P. Doherty, J. Mylopoulos, C.A. Welty (Eds.), Proc. 10th Int. Conf. on Principles of Knowledge Representation and Reasoning, Lake District, UK, AAAI Press, 2006, pp. 463–471.
[56] M. Sipser, Introduction to Theory of Computation, second ed., Thomson Course Technology, Boston, MA, 2005.
[57] D. Smith, D. Weld, Conformant graphplan, in: J. Mostow, C. Rich (Eds.), Proc. 15th National Conf. on Artificial Intelligence, Madison, WI, AAAI Press/MIT Press, 1998, pp. 889–896.
[58] E. Sondik, The optimal control of partially observable Markov decision processes over the infinite horizon: Discounted costs, Oper. Res. 26 (2) (1978).
[59] L.J. Stockmeyer, The polynomial time hierarchy, Theoret. Comput. Sci. 3 (1977) 1–22.
[60] Time, And now, Master Mind, Time 106 (22) (December 1, 1975) 73.
[61] H. Turner, Polynomial-length planning spans the polynomial hierarchy, in: S. Flesca, I. Giovambattista (Eds.), Proc. 8th European Conf. on Logics in Artificial Intelligence, Cosenza, Italy, in: Lecture Notes in Comput. Sci., vol. 2424, Springer, 2002, pp. 111–124.
[62] A.C.-C. Yao, Separating the polynomial-time hierarchy by oracles, in: Proc. 26th Annual Symp. on Foundations of Computer Science, Portland, OR, IEEE Press, 1985, pp. 1–10.

Please cite this article in press as: B. Bonet, Conformant plans and beyond: Principles and complexity, Artificial Intelligence (2009), doi:10.1016/j.artint.2009.11.001