To appear, *Proc. 14th National Conf. on AI (AAAI-97)*, Providence, August, 1997

Structured Solution Methods for Non-Markovian Decision Processes*

Fahiem Bacchus
Dept. of Computer Science
University of Waterloo
Waterloo, Ontario
Canada, N2L 3G1
fbacchus@logos.uwaterloo.ca

Craig Boutilier
Dept. of Computer Science
University of British Columbia
Vancouver, B.C.
Canada, V6T 1Z4
cebly@cs.ubc.ca

Adam Grove
NEC Research Institute
4 Independence Way
Princeton NJ 08540, USA
grove@research.nj.nec.com

Abstract

Markov Decision Processes (MDPs), currently a popular method for modeling and solving decision theoretic planning problems, are limited by the Markovian assumption: rewards and dynamics depend on the current state only, and not on previous history. *Non-Markovian* decision processes (NMDPs) can also be defined, but then the more tractable solution techniques developed for MDP's cannot be directly applied. In this paper, we show how an NMDP, in which temporal logic is used to specify history dependence, can be automatically converted into an equivalent MDP by adding appropriate *temporal variables*. The resulting MDP can be represented in a structured fashion and solved using structured policy construction methods. In many cases, this offers significant computational advantages over previous proposals for solving NMDPs.

1 Introduction

Markov decision processes (MDPs) have proven to be an effective modeling and computational paradigm for decision-theoretic planning (DTP). MDPs allow one to deal with planning problems that involve uncertainty, multiple objectives, and nonterminating (process-oriented) behavior. A fundamental assumption of this model is that the reward function and system dynamics of the underlying process are *Markovian*—all the information needed to determine the value of a particular state or the effects of an action at that state must be encoded within the state itself. This allows computationally effective dynamic programming techniques to be used to solve decision problems [Put94].

Nevertheless, the Markovian requirement is often not met by planning problems that are encoded in the "obvious" way. For instance, it is often natural to specify desirable behaviors by referring to trajectory properties (properties of the sequence of states passed through, i.e., the system's history) in addition to just the current state. This has shown up in work on planning [HH92, Dru89, Kab90, GK91] (e.g., in the use of maintenance goals); and in [BBG96] we have argued that many reward functions for *process-oriented* problems are most appropriately viewed in this light (e.g., responses to requests, maintaining desirable systems properties, etc.). For instance, rewarding an agent for achieving a goal within k steps of a request being issued is a natural, yet history-dependent, specification of desirable behavior. Similarly, process dynamics (action effects) are sometimes most naturally expressed in a history dependent fashion.

In [BBG96] we examined *Non-Markovian decision processes* (NMDPs) and identified two key issues, namely, the specification of non-Markovian properties and the solution of NMDPs.[2] A temporal logic called PLTL was used as a mechanism for specifying the non-Markovian aspects of a system, and we will adopt the same approach here. However, we propose a very different way of solving the NMDPs so specified. As in the earlier paper, we develop a method for automatically converting an NMDP into an *equivalent* MDP, solutions to which can be re-interpreted to yield solutions to the original NMDP. The key difference between the two papers is that [BBG96] presents a *state-based* construction, while this paper works with *structured* representations of (N)MDPs. Each approach has advantages and disadvantages.

In either case, one can think of each state in the original NMDP as leading to multiple states in the resulting MDP, the new states being distinguished by various relevant *histories*. The difference, in essence, is how the new MDP is represented. The algorithm in [BBG96] takes an NMDP and a reward function specified using PLTL formulas, and produces a new MDP whose states are listed explicitly. One advantage of this approach (which will not be true of the proposal in the current paper) is that it is possible to produce a *minimal* expanded MDP (i.e., one with the fewest number of states necessary to capture the required historical distinctions). However, as we now discuss, there are also important disadvantages inherent in state-based constructions.

To understand these disadvantages, note that many DTP problems are described in terms of a set of variables or propositions (particularly in AI where logical representations are common). The resultant MDPs have as states all possible assignments of values to these variables. In such cases, a major difficulty with any state-based algorithm for MDPs is that

* The work of Fahiem Bacchus and Craig Boutilier was supported by the Canadian government through their NSERC and IRIS programs.

[2] In [BBG96] we considered non-Markovian reward functions only, although the approach could easily be extended to deal with history dependence in the system dynamics as well. In this paper we consider both rewards and dynamics explicitly.

fact that the state space grows exponentially with the number of problem variables. A recent focus in DTP research has been the development of MDP representation and solution techniques that do not require an explicit enumeration of the state space. For instance, the use of STRIPS [BD94, DF95, KHW94] or Bayes nets [BDG95, BD96] to represent actions in MDPs, and *structured policy construction* (SPC) methods that exploit such representations [DF95, BDG95, BD96] to avoid explicit state-based computations when solving MDPs, promise to make MDPs more effective for such DTP problems.

For such problems, the NMDP conversion algorithm proposed in [BBG96] has some obvious drawbacks. First, being state-based, its complexity is exponential in the number of problem variables, because even if the original NMDP has a compact representation in terms of variables the final MDP will not. Second, since the MDP output by that algorithm is represented as a set of states, any structure present in the original (structured) NMDP will be lost or obscured, preventing the use of SPC algorithms. Finally, as we will see, the history that must be encoded itself exhibits significant structure, but this is not exploited in [BBG96].

Our approach to NMDP conversion in this paper assumes the existence of a compact variable-based description of the given NMDP, and works by adding *temporal variables* to this description. We thus expand the state space without regard to minimality, but this expansion is *implicit*—we never enumerate all states. The the conversion is thus very efficient, and retains the structured and compact representation we started with. The output, being a structured MDP, can be used directly by SPC algorithms. This last consideration has an interesting and fortunate consequence. Unlike the state-based approach, our proposal here does not produce an MDP whose (implicit) state space is minimal; but to compensate for this, we can exploit the fact that SPC algorithms have some ability to automatically (and dynamically) detect the relevance of particular variables at various points in policy construction. As such, unlike [BBG96], we need not be directly concerned about only adding the "relevant" history. To a significant extent, relevance can be detected during optimization by appropriate SPC algorithms.

This points to a related advantage of our technique, which is that relevant history is dynamically determined in SPC algorithms. The algorithm of [BBG96] is based on a *static analysis* of the NMDP, before optimal policy construction is attempted. That algorithm must encode enough history at a given state s to determine the reward at *any* future reachable state t. This ensures that any policy, and not just the optimal one, will have the same total reward in the constructed MDP as in the original NMDP. But in fact we are generally only interested in the optimal policy. As we construct this policy, we may notice that many *a priori* reachable states t (i.e., reachable with non-zero probability through *some* sequence of actions) are, in fact, not reachable when the optimal policy is adopted. Consequently, some of the history we added (in order to determine the reward at state t) may end up being "irrelevant". Our algorithm, in conjunction with SPC algorithms used for optimization, implements a *dynamic analysis* of the NMDP. Once it is determined that t is not reachable during policy construction, history relevant only to t (and related states) will be ignored. Though the state space is larger, only relevant distinctions are, for the most part, considered.

We conjecture that such "dynamic irrelevance" is especially likely to arise in circumstances involving temporally dependent rewards. For example, a temporally-extended reward function may associate a reward with delivering a item within 5 time stages of an order being placed. Under certain conditions (e.g., inclement weather) the risk associated with any attempt at delivery may be too great for such an undertaking. During policy construction, this will be recognized and the history relevant to determining whether this goal can be achieved (i.e., keeping track of when in the past an order was placed) can be ignored in such states. The "irrelevance" of the required history cannot be detected *a priori*.

A final contrast with [BBG96] is the relative simplicity of our proposal in the current paper, which is based on well-known properties of temporal logic and the observation that these can be made to integrate well with SPC. Although the technical aspects of our contribution are straightforward, it has considerable potential for improving our ability to solve history-dependent decision problems.

We begin with a brief description of MDPs, NMDPs and the temporal logic used in [BBG96] to specify trajectory properties. We also give an overview of the SPC algorithm of [BDG95]. Then we present our technique of adding temporal variables to convert an NMDP to an MDP. We close the paper with some further observations.

2 Background

2.1 Markov Decision Processes

A *fully observable Markov Decision Process* [How60, Put94] can be characterized by a finite set of states S, a set of actions A, and a reward function R. The actions are characterized by probability distributions, and we write $\Pr(s_1, a, s_2) = p$ to denote that s_2 is reached with probability p when action a is performed in state s_1. Full observability entails that the agent always knows what state it is in. A real-valued *reward function* R reflects the objectives, tasks and goals to be accomplished by the agent, with $R(s)$ denoting the (immediate) utility of being in state s. Thus, an MDP consists of S, A, R and the set of transition distributions $\{\Pr(s, a, \cdot) \mid a \in A; s \in S\}$.

A *stationary Markovian policy* is a mapping $\pi : S \rightarrow A$, where $\pi(s)$ denotes the action an agent should perform whenever it is in state s. We adopt *expected total discounted reward* over an infinite horizon as our optimality criterion: the current value of future rewards is discounted by some factor β ($0 < \beta < 1$), and we maximize the expected accumulated discounted rewards over an infinite time period. The expected value of a fixed policy π at any state s can be shown to satisfy [How60]:

$$V_\pi(s) = R(s) + \beta \sum_{t \in S} \Pr(s, \pi(s), t) \cdot V_\pi(t)$$

The value of π at any state s can be computed by solving this system of linear equations. A policy π is *optimal* if $V_\pi(s) \geq V_{\pi'}(s)$ for all $s \in S$ and policies π'. V_π is called the *value*

function. We refer to [Put94] for an excellent treatment of MDPs and associated computational methods. See [DW91, BD94, BDG95] on the use of MDPs for DTP.

2.2 Structured Representations

For DTP problems, we are often interested in MDPs that can be represented concisely in terms of a set of features or variables. We adapt the Bayes net/decision tree representation used by [DK89, BDG95], representing an MDP with a set of variables $\mathbf{P}$, a reward decision tree Tree_R, the set of actions A, and a set of action decision trees $\{\mathit{Tree}(a,p) \mid a \in A; p \in \mathbf{P}\}$.

$\mathbf{P}$ is a set of n variables that implicitly specifies the state space S. For simplicity, we assume that all variables take two values, $\{\top$ (true), $\bot$ (false)$\}$; consequently we can identify each variable $p \in \mathbf{P}$ as a boolean proposition. The state space S is then the set of all possible truth assignments to these variables (i.e., S is the product space of the variable domains). The number of states is exponential in the number of variables (thus, in our case, 2^n).

The rest of the representation relies on the use of decision trees whose internal nodes are tests on individual variables in $\mathbf{P}$. The leaves of these trees are labeled with different types of values, depending on the type of tree. Any of these trees, say *Tree*, can be applied to a state s, to yield a value $\mathit{Tree}[s]$. This value is computed by traversing the tree using the truth assignments specified by s to determine the direction taken at internal nodes. The value returned is the label of the leaf node reached. That is, $\mathit{Tree}[s]$ is the label of the leaf associated with the (unique) branch of the tree whose variable labels are consistent with the values in s.

Tree_R is a decision tree that specifies the (immediate) reward of each state in S. In particular, the leaves of Tree_R are labeled with real values, and the reward $R(s)$ assigned to any state s is $\mathit{Tree}_R[s]$. The set of action trees determine transition probabilities. These trees have their leaves labeled with real numbers in $[0,1]$. $\mathit{Tree}(a,p)[s]$ specifies the probability that p will be true in the next state given that action a was executed in state s. Thus for each variable $p' \in \mathbf{P}$ we can determine the probability of it being true in the next state from $\mathit{Tree}(a,p')[s]$. These events are assumed to be independent so we can obtain the entire distribution over the successor states, $\Pr(s,a,\cdot)$, by simply multiplying these probabilities.[3]

Although [BDG95] describes their representation in terms of Bayes nets, it is nevertheless equivalent to ours, including the independence assumption. One advantage of the Bayes net representation is that it suggests an easy way of relaxing this assumption. (This possibility was not explored in [BDG95], but see [Bou97] for details.) Thus, SPC algorithms can be modified to deal with dependence between present variables, although they become somewhat more complex. Such modifications are entirely compatible with our proposals here.

[3] In particular, we have

$$\Pr(s,a,s') = \prod_{p\,:\,s' \models p} \mathit{Tree}(a,p)[s] \prod_{p\,:\,s' \not\models p} (1 - \mathit{Tree}(a,p)[s]).$$

2.3 Structured Policy Construction

The basic idea behind the SPC algorithms described in [BDG95, BD96] is the use of tree-structured representations during policy construction. In particular, a policy π can be represented as a decision tree Tree_π where the leaves are labeled by actions. That is, $\mathit{Tree}_\pi[s]$ specifies the action to execute in state s. Similarly a value function V can be represented as a tree with real-valued labels.

The SPC algorithm is based on the observation that some of the traditional algorithms for constructing optimal policies can be reformulated as tree-manipulation algorithms. In particular, at all points in the algorithm the current value function and current policy are represented as trees. Intuitively, these algorithms dynamically detect the relevance of particular variables, under specific conditions, to the current value function or policy. The particular tree-manipulation steps necessary are not trivial, but the details are not directly relevant here (they are presented in [BDG95]).

We have already alluded to some of SPC's properties. If the dynamics and reward function of the MDP are simple, the optimal policy very often has a simple structure as well (see [BDG95] for examples). SPC will find this policy in just as many iterations as modified policy iteration (a popular and time-efficient algorithm), but often without ever considering "large" trees—even though it is (implicitly) optimizing over exponentially many states. Variables that are not relevant, or are only relevant in some contexts, will not necessarily have an adverse effect on complexity—in many cases, they just do not appear as decision nodes in the trees when they are not needed. We note that SPC offers another advantage, namely its amenability to approximation; see [BD96].

2.4 Non Markovian Decision Processes

Following [BBG96], we use a temporal logic called PLTL (Past Linear Temporal Logic) to specify the history dependence of rewards and action effects. PLTL is a past version of LTL [Eme90]. We assume an underlying finite set of propositional constants $\mathbf{P}$, the usual truth functional connectives, and the following temporal operators: S (since), $\boxminus$ (always in the past), $\diamondminus$ (once, or sometime in the past) and $\ominus$ (previously).[4] The formulas $\phi_1 \mathsf{S} \phi_2$, $\boxminus\phi_1$, $\diamondminus\phi_1$ and $\ominus\phi_1$ are well-formed when ϕ_1 and ϕ_2 are.[5] The semantics of PLTL is described with respect to models of the form $T = \langle s_0, \cdots, s_n\rangle$, $n \geq 0$, where each s_i is a *state* or truth assignment over the variables in $\mathbf{P}$. For any *trajectory* $T = \langle s_0, \cdots, s_n\rangle$, and any $0 \leq i \leq n$, let $T(i)$ denote the *initial segment* $T(i) = \langle s_0, \cdots, s_i\rangle$.

A temporal formula is true of $T = \langle s_0, \cdots, s_n\rangle$ if it is true at the last (or *current* state) with respect to the history reflected in the trajectory. We define the truth of formulas inductively as follows:

- $T \models P$ iff $s_n \models P$, for $P \in \mathbf{P}$

[4] These are the backward analogs of the LTL operators until, always, eventually and next, respectively.

[5] We use the abbreviation $\ominus^k$ for k iterations of the $\ominus$ modality (e.g., $\ominus^3\phi \equiv \ominus\ominus\ominus\phi$), and $\ominus^{\leq k}$ to stand for the disjunction of $\ominus^i$ for $1 \leq i \leq k$, (e.g., $\ominus^{\leq 2}\phi \equiv \ominus\phi \vee \ominus\ominus\phi$).

- $T \models \phi_1 \wedge \phi_2$ iff $T \models \phi_1$ and $T \models \phi_2$
- $T \models \neg\phi$ iff $T \not\models \phi$
- $T \models \phi_1 \mathsf{S} \phi_2$ iff there is some $i \leq n$ s.t. $T(i) \models \phi_2$ and for all $i < j \leq n$, $T(j) \models \phi_1$ (that is, ϕ_2 was true sometime in the past and ϕ_1 has been true since then)
- $T \models \boxminus\phi$ iff for all $0 \leq i \leq n$, $T(i) \models \phi$ (ϕ has been true at each point in the past)
- $T \models \diamondminus\phi$ iff for some $0 \leq i \leq n$, $T(i) \models \phi$ (ϕ was true at some point in the past)
- $T \models \ominus\phi$ iff $n > 0$ and $T(n-1) \models \phi$ (ϕ was true at the previous state)

If, for example, we wanted to reward behaviors that achieve a condition G immediately after it is requested by a command C, we could specify a reward function that rewards states satisfying the PLTL formula $G \wedge \ominus C$. Similarly, rewarding states satisfying $G \wedge \ominus^{\leq k} C$ would reward behaviors that achieve G within k steps its request. [BBG96] gives further examples of how the logic can be used to specify useful historical dependencies. Using a logic for making such specifications provides all of the usual advantages gained from logical representations. In particular, we have a compositional language that can be used to express arbitrarily complex specifications, and the same time we have a precise semantics for our specification no matter how complex.

Our proposal for using PLTL formulas within the tree-structured representational framework of Section 2.2 is straightforward. Recall that we represented the reward function and dynamics using decision trees, whose tests were on the value of individual variables. We simply extend this by allowing a internal node in the decision tree to also test the value of an arbitrary PLTL formula. In this way, both rewards and dynamics can depend on history, and we can represent any NMDP, whose history dependence is specified using PLTL, as a structured NMDP.

3 Temporal Variables

Given some structured NMDP $\mathcal{N}$, as described above, we want to find an equivalent[6] structured MDP $\mathcal{M}$, so that the SPC algorithm can be used to find optimal policies. To do this, we introduce *temporal variables* into the domain that refer to relevant properties of the current trajectory. Each temporal variable is used to track the truth of some *purely temporal formula* of PLTL (i.e., a formula whose main connective is temporal). Temporal variables are thus boolean.

3.1 Conditions on Temporal Variable Sets

In the new MDP $\mathcal{M}$, the state must contain sufficient information to determine rewards and the dynamics; unlike $\mathcal{N}$, it will not be able to refer to history explicitly. Consider any temporal formula ϕ appearing as a decision node in one of the decision trees of $\mathcal{N}$. It follows that any state in $\mathcal{M}$ must contain enough information to decide if ϕ is true. For this reason, we must at least add temporal variables to the state description that provide *sufficient history* so that this information can be determined. Let $\mathbf{T}$ be the added set of temporal variables. Each of these variables is associated with a formula of PLTL, and we will often treat these variables as if they were formulas.

Definition 3.1 *A set of temporal variables* $\mathbf{T}$ *is* sufficient *for* ϕ *iff there is some boolean function* b_ϕ *such that* $\phi \equiv b_\phi(\mathbf{T}, \mathbf{P})$. ∎

In other words, $\mathbf{T}$ is sufficient for ϕ if the truth of ϕ is determined by some subset of the elements of $\mathbf{T}$, together with knowledge of certain state variables.

Since any formula in PLTL has the form $b(t_1, \cdots, t_k)$ for some boolean function b and purely temporal formulas $t_1, \cdots, t_k$, finding a sufficient set of temporal formulas is trivial: we can simply strip the main boolean connectives from a temporal formula ϕ until only purely temporal formulas remain, then make each of these a temporal variable.

However this set of temporal variables turns out not to be suitable, because sufficiency is not the only requirement for the set of variables $\mathbf{T}$ needed to build a suitable MDP. Since the temporal variables we choose will be incorporated into the new MDP $\mathcal{M}$, we must be able to determine the dynamics of these new variables. That is, just as we have trees *Tree*(a, p) specifying the dynamics of the state variables $p \in \mathbf{P}$ we must be able to construct trees *Tree*(a, t) that tell us how to update the temporal variables $t \in \mathbf{T}$. Furthermore, these dynamics must be Markovian.

Definition 3.2 *A set of temporal variables* $\mathbf{T}$ *is* dynamically closed *iff, for each* $t \in \mathbf{T}$*, there is some boolean function* b_t *such that* $t \equiv \ominus b_t(\mathbf{T}, \mathbf{P})$. ∎

Given dynamic closure, the truth of any variable in $\mathbf{T}$ at a given state can be determined from the values of variables in $\mathbf{T}$ and $\mathbf{P}$ at the *previous* state. Hence, dynamic closure ensures that we can update the temporal variables while still satisfying the Markov assumption.

Let $\mathbf{T}$ be dynamically closed, and consider b_t as defined above. Like any boolean formula, b_t can be evaluated using some decision tree T_t; the internal nodes of T_t test the values of variables in $\mathbf{T} \cup \mathbf{P}$ and the leaves are labeled $\top$ or $\bot$.[7] The dynamics of any temporal variable t are now easy to define: we can set *Tree*(a, t), for all actions a, to be T'_t, where T'_t is the tree exactly like T_t except that where T_t evaluates to $\top$ (resp., $\bot$) T'_t evaluates to 1 (resp., 0); that is, the truth values are translated into probabilities.

We note that the dynamics of the temporal variables are independent of the action executed. Furthermore, the determinism of *Tree*(a, t) ensures that our independence assumptions on action effects (Section 2.2) remain valid.

We have shown that, if we augment the set of variables of $\mathcal{N}$ with a dynamically closed set of temporal variables $\mathbf{T}$ and define the dynamics as just discussed, then the resulting

[6] Intuitively, equivalence implies that an optimal policy for the constructed MDP can be re-interpreted as an optimal policy for the original NMDP. See [BBG96] for a formal definition.

[7] In general, there are many trees that can be used to evaluate the formula b_t; an arbitrary tree can be chosen. However, some may be more compact (hence, lead to greater efficiency) than others.

NMDP is equivalent to the original with the exception of the added variables, and the dynamics of the temporal variables are Markovian. However, an even more important feature of the construction is the following: because of the way we have defined the dynamics, in any valid trajectory all temporal variables in $\mathbf{T}$ will *faithfully* reflect their intended meaning. More precisely, consider any trajectory through the state space of the new MDP. Since each $t \in \mathbf{T}$ is one of the variables defining the state space, then at each point t will have some value ($\top$ or $\bot$). But t is also a logical formula, and so we can also ask whether (formula) t is *true* or *false* of the trajectory we have followed. Our construction ensures that (variable) t is true at any state iff the corresponding formula is true of the trajectory followed (assuming all variables are given correct values at time 0, the initial state). In other words, a variable's *value* coincides with its *truth*.

Suppose $\mathbf{T}$ is both dynamically closed, and sufficient for every temporal formula ϕ that appears as a decision node either in Tree_R or in any $\mathit{Tree}(a, p)$ in $\mathcal{N}$; for concreteness, imagine that ϕ appears in Tree_R. Sufficiency implies we can replace the test of ϕ by a test on the values of $\mathbf{T} \cup \mathbf{P}$, using b_ϕ as defined in Definition 3.1. In fact, we can find some decision tree that evaluates b_ϕ, and substitute this tree for the internal nodes in $\mathcal{N}$ that test ϕ. Doing this for all such ϕ, the result is a different tree for evaluating rewards (or in the case of $\mathit{Tree}(a, p)$, dynamics) that is semantically equivalent to the original, and tests only variables in $\mathbf{T} \cup \mathbf{P}$. Thus, by using the enlarged set of variables, we have removed all tests that depend explicitly on history: this construction has delivered an MDP $\mathcal{M}$, equivalent to $\mathcal{N}$, as required. Of course, the key to the correctness of this construction is the faithfulness property.

3.2 Construction of Temporal Variable Sets

It remains to show that a suitable set $\mathbf{T}$ exists; in other words, that for any set of PLTL formulas Φ we can construct a dynamically closed set of temporal variables that is sufficient for all $\phi \in \Phi$.

To begin, we define a *subformula* of a PLTL formula ϕ to be any well-formed PLTL formula contained as a substring of ϕ. For example, if $\psi = p \wedge \boxminus (q \vee \ominus \neg p)$ then the subformulas of ψ are ψ itself, p, $\boxminus (q \vee \ominus \neg p)$, $q \vee \ominus \neg p$, q, $\ominus \neg p$, and $\neg p$. A first proposal for $\mathbf{T}$ might be to consider the set of all purely temporal subformulas of $\phi \in \Phi$, denoted $\mathit{PTSub}(\Phi)$. For ψ above these are the subformulas $\boxminus (q \vee \ominus \neg p)$ and $\ominus \neg p$. This ensures sufficiency simply because any $\phi \in \Phi$ is a boolean combination of propositional symbols and its purely temporal subformulas. (In fact, we would not need all subformulas: for ψ above, only $\boxminus (q \vee \ominus \neg p)$ would be needed if sufficiency were all we cared about.)

The problem with this is that it does not satisfy the dynamic closure property. For instance, the truth of $\boxminus (q \vee \ominus \neg p)$ depends not only on what happened in the past, but also on whether $q \vee \ominus \neg p$ is true now. (Recall that the semantics of $\boxminus$ are, informally, "…has been true always in the past, up to *and including* the present.") In this case, the truth of $\ominus \neg p$ in the *present* must be known. Our representation, however, requires that a variable's value can be determined by the *previous* values of other variables.[8]

Instead, we will construct a set of temporal variables that have $\ominus$ as their main connective, because the truth of such a formula depends only on the past. In fact, as we now show, we can define $\mathbf{T}$ by prepending $\ominus$ to each subformula in $\mathit{PTSub}(\Phi)$ except those that already begin with $\ominus$ (these are retained without change). That is:

$$\begin{aligned} \mathbf{T} \;=\; & \{\ominus \psi \mid \psi \in \mathit{PTSub}(\Phi), \psi \neq \ominus \xi\} \\ & \cup \{\ominus \psi \mid \ominus \psi \in \mathit{PTSub}(\Phi)\} \end{aligned}$$

To show sufficiency, we use the following equivalences of PLTL that allow us to insert preceding $\ominus$ operators before any other modal operator:

1. $\alpha \,\mathsf{S}\, \beta \;\equiv\; \beta \vee (\alpha \wedge \ominus(\alpha \,\mathsf{S}\, \beta))$.
2. $\boxminus \alpha \;\equiv\; \alpha \wedge \ominus(\boxminus \alpha)$.
3. $\diamondminus \alpha \;\equiv\; \alpha \vee \ominus(\diamondminus \alpha)$.

Consider any $\phi \in \Phi$; ϕ is, of course, a boolean combination of primitive propositions in $\mathbf{P}$ and purely temporal subformulas of PLTL. If any of the temporal subformulas used in this expression do not begin with $\ominus$, we can replace them by the equivalent expressions according to the above. This replacement process may need to be repeated several times, but eventually it will terminate giving us a boolean combination over $\mathbf{P} \cup \mathbf{T}$ that is equivalent to ϕ, as required. Note that the same construction allows us to represent *any* subformula of ϕ as a boolean combination over $\mathbf{P} \cup \mathbf{T}$.

Dynamic closure is shown very simply. Let $\ominus \psi \in \mathbf{T}$. By definition ψ must be a subformula of some $\phi \in \Phi$ and so, as we have just seen, is equivalent to some boolean combination b_ψ over $\mathbf{P} \cup \mathbf{T}$. Thus $\ominus \psi \;\equiv\; \ominus b_\psi$ as required for dynamic closure.

As an illustration, consider $\phi = \diamondminus(p \,\mathsf{S}\, (q \vee \ominus r)$. From this, we form the temporal variable set

$$\mathbf{T} = \{t_1 = \ominus\diamondminus(p \mathsf{S}(q \vee \ominus r), t_2 = \ominus(p \mathsf{S}(q \vee \ominus r), t_3 = \ominus r)\}.$$

Then ϕ can be decomposed as follows:

$$\begin{aligned} & \diamondminus(p \,\mathsf{S}\, (q \vee \ominus r) \\ & \quad \equiv \quad p \,\mathsf{S}\, (q \vee \ominus r) \vee \ominus\diamondminus(p \,\mathsf{S}\, (q \vee \ominus r) \\ & \quad \text{i.e.} \quad p \,\mathsf{S}\, (q \vee \ominus r) \vee t_1 \\ & \quad \equiv \quad ((q \vee \ominus r) \vee (p \wedge \ominus(p \,\mathsf{S}\, (q \vee \ominus r)))) \vee t_1 \\ & \quad \text{i.e.} \quad ((q \vee t_3) \vee (p \wedge t_2)) \vee t_1 \end{aligned}$$

4 Concluding Observations

We have shown how, given a structured NMDP consisting of reward and action representations involving PLTL formulas, one can construct a set of temporal variables that is dynamically closed and sufficient for those formulas, and use these

[8] Again we note that, in principle, dependence on present variables can be used; but we retain the original formulation for reasons discussed in Section 2.2.

to specify an equivalent MDP in compact form. We conclude with some observations about our proposal and some directions for future work.

We first note that the new MDP can be constructed efficiently. Let Φ be the collection of PLTL formulas appearing in the NMDP specification. It is easy to see that the number of temporal variables added is at most equal to the number of purely temporal subformulas contained in Φ, which is bounded by the total number of temporal operators in Φ. Thus, we are adding only a modest number of new variables. The time required to do so is $O(T|\Phi|)$, where T is the total size of the reward and action trees and $|\Phi|$ is the sum of the lengths of formulas in Φ. Of course, one must keep in mind that the implicit state space grows exponentially in the number of added variables. But as we discussed in Sections 1 and 2.3, the size of the implicit state space is not necessarily the key factor in the complexity of SPC algorithms.

We do not claim that our construction adds the minimal number of extra variables to achieve its purpose. In fact, it is easy to see that minimal state space size is sometimes not achievable by adding variables at all. The reason is that the required history can vary from state to state (indeed, this fact motivates much of [BBG96]). Our approach cannot make such fine distinctions: every state has the same set of variables "added" to it. On the other hand, the "dynamic irrelevance" feature of SPC may compensate for this. That is, although the temporal variables can *potentially* cause us to make unnecessary distinctions between histories at certain states (as compared to [BBG96]), SPC attempts to focus only on the variables whose values influence the choices made by the optimal policy, and so will avoid some of these irrelevant distinctions. Furthermore, SPC can avoid "dynamic irrelevances" that cannot be detected by the approach of [BBG96], and is much more amenable to approximation in the solution of the resulting MDP. Empirical studies are, of course, needed to quantify this tradeoff. We suspect that there will be a range of domains where the SPC approach we are suggesting here will be superior to the state-space based approach of [BBG96] and vice versa. The interesting question will be to attempt to characterize domain features that tend to favor one approach over the other.

Not surprisingly, NMDPs (even in the structured framework we are considering) can be much harder to solve than a comparably sized MDP. As a simple example, suppose one gets a reward each time $q \wedge \ominus^n p$ is true. Suppose also that there is an action a that, if taken, is likely to lead to q becoming true at some (unpredictable) time within the next n steps. The optimal policy could well need to keep track of exactly when p was true among the previous n steps (because it has to know whether and when to try to achieve q). There may be no sub-exponential (in n) decision tree representation of such a policy. Hence, SPC is not guaranteed to be efficient.

Finally, in this paper, we have omitted any discussion of the various optimizations that are possible. For instance, one might gain by carefully choosing the decision tree used to evaluate b_ϕ and b_t (Section 3). These issues deserve careful study. More importantly for future work, however, is the need for empirical studies to explore the actual performance that might be seen in practical cases.

References

[BBG96] Fahiem Bacchus, Craig Boutilier, and Adam Grove. Rewarding behaviors. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 1160–1167, Portland, OR, 1996.

[BD94] Craig Boutilier and Richard Dearden. Using abstractions for decision-theoretic planning with time constraints. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 1016–1022, Seattle, 1994.

[BD96] Craig Boutilier and Richard Dearden. Approximating value trees in structured dynamic programming. In *Proceedings of the Thirteenth International Conference on Machine Learning*, pages 54–62, Bari, Italy, 1996.

[BDG95] Craig Boutilier, Richard Dearden, and Moisés Goldszmidt. Exploiting structure in policy construction. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pages 1104–1111, Montreal, 1995.

[Bou97] Craig Boutilier. Correlated action effects in decision-theoretic regression. (manuscript), 1997.

[DF95] Thomas G. Dietterich and Nicholas S. Flann. Explanation-based learning and reinforcement learning: A unified approach. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 176–184, Lake Tahoe, 1995.

[DK89] Thomas Dean and Keiji Kanazawa. A model for reasoning about persistence and causation. *Computational Intelligence*, 5(3):142–150, 1989.

[Dru89] M. Drummond. Situated control rules. In *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning*, pages 103–113, Toronto, 1989.

[DW91] Thomas Dean and Michael Wellman. *Planning and Control*. Morgan Kaufmann, San Mateo, 1991.

[Eme90] E. A. Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, Volume B*, chapter 16, pages 997–1072. MIT, 1990.

[GK91] P. Godefroid and F. Kabanza. An efficient reactive planner for synthesizing reactive plans. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, pages 640–645, 1991.

[HH92] Peter Haddawy and Steve Hanks. Representations for decision-theoretic planning: Utility functions for deadline goals. In *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning*, pages 71–82, Cambridge, 1992.

[How60] Ronald A. Howard. *Dynamic Programming and Markov Processes*. MIT Press, Cambridge, 1960.

[Kab90] F. Kabanza. Synthesis of reactive plans for multi-path environments. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 164–169, 1990.

[KHW94] Nicholas Kushmerick, Steve Hanks, and Daniel Weld. An algorithm for probabilistic least-commitment planning. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 1073–1078, Seattle, 1994.

[Put94] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley, New York, 1994.