A More Expressive Behavioral Logic for Decision-Theoretic Planning

Charles Gretton

NICTA, Canberra, Australia; Australian National University; Griffith University

Abstract. We examine the problem of compactly expressing models of non-Markovian reward decision processes (NMRDP). In the field of decision-theoretic planning NMRDPs are used whenever the agent's reward is determined by the history of visited states. Two different propositional linear temporal logics can be used to describe execution histories that are rewarding. Called PLTL and \$FLTL, they are backward and forward looking logics respectively. In this paper we find both to be expressively weak and propose a change to \$FLTL resulting in a much more expressive logic that we have called $\$^* FLTL$. The time complexities of $\$^* FLTL$ and \$FLTL related model checking operations performed in planning are the same.

Keywords: decision-theoretic planning, non-Markovian, temporal logic

1 Introduction

Problems in robot navigation, elevator control, energy distribution networks, *etc.* can often be modelled as stochastic decision processes. In the field of decision-theoretic planning we research general computational tools for synthesizing policies that efficiently and/or robustly control such processes. The *de facto* standard process model is the fully observable finite state propositional Markov Decision Process (MDP) described by Boutilier et al. (1999). That model is Markovian in the sense that the current state of the process determines the instantanious effects of an action and reward allocated. Reality motivates a richer formalism, one that can compactly model processes where reward allocation occurs when the system's behavior (i.e. the execution history) exhibits desirable temporally extended properties. Common *rewarding behaviors* include the achievement of an objective/configuration once and only once, the maintenance of a desirable configuration, the periodic achievement of an objective, and the achievement of a dynamic objective. A decision process in which rewards depend on the sequence of states passed through rather than merely on the current state is called a decision process with *non-Markovian rewards* (NMRDP). There have been a number of formalisms proposed to compactly describe NMRDP. A detailed review and empirical study of all existing solution methods and related formalisms is given by Thiébaux et al. (2006).

Proposals to date describe rewarding behaviors using one of two linear temporal logistics, either PLTL or \$FLTL.[1] In this paper we review propositional MDPs, these two temporal logics, the motivations for each in our setting, and discuss how NMRDPs are described in practice. We also briefly review the different approaches to solving NMRDPs. We characterise rewarding behaviors in terms of language, and prove that PLTL and \$FLTL can only describe rewarding behaviors that correspond to *noncounting* regular languages. We then make a minor change to \$FLTL yielding a much more expressive logic that is able to describe behaviors that correspond to counting languages. Examples of counting languages in everyday life occur in the lighting sequences at traffic intersections – Consider a first car arrives, the signal facing that vehicle remains red for a fixed count of states, is set to green for a fixed count of states, then orange, and so on. In mobile robotics, rewarding battery-charge profiles correspond to counting languages. Importantly, both $\$^*FLTL$ and \$FLTL *progression* – i.e. the operation used in planning to reason about rewarding behaviors – have the same computational time complexity.

2 Propositionally Factored Markov Decision Processes

An MDP is given by a four-tuple $\langle \mathcal{S}, \mathcal{A}, Pr, \mathsf{R} \rangle$. $\mathcal{S}$ is a finite set of states. $\mathcal{A}$ is a finite set of actions. Where $s, s' \in \mathcal{S}$, $a \in \mathcal{A}$ then $Pr(s, a, s')$ is the probability of a transition from state s to s' given action a is executed at state s. Also present is a bounded real-valued reward function $\mathsf{R} : \mathcal{S} \rightarrow \mathbb{R}$. R is bounded in the sense that there is a positive constant c so that for all $s \in \mathcal{S}$, $|\mathsf{R}(s)| < c$.

A *flat* MDP model, i.e. where the atoms of the model are states and actions, is a fundamental conceptual tool in the study of automated planning, however it does not expose how process models are described. Typically (see Boutilier and Dearden, 1994) the underlying process dynamics are described in terms of a finite set of probabilistic STRIPS operators $\mathcal{A}$ over state-characterising propositions $\mathcal{P}$. A problem state s is characterised by a set of propositions, in the sense that $s \subseteq \mathcal{P}$. For $p \in s$ we say proposition p is true in s. The reward description however is usually flat, tabulating $\mathsf{R}(s)$ for each s with non-zero reward.

Describing the process dynamics in this setting, every stochastic action has a precondition $poss(a)$, which is a set of propositions. An action can be executed at a state s when $poss(a) \subseteq s$. Let $\mathcal{A}(s)$ be the set of actions that can be executed at state s. When $a \in \mathcal{A}(s)$ is executed, nature decides amongst a small set of deterministic outcomes $\mathcal{O}(a) \equiv \{\mathtt{da}_1, \ldots, \mathtt{da}_k\}$ what actually occurs. Each possible outcome is described as a deterministic STRIPS operator with no precondition. To keep this exposition simple, for any two distinct actions $a_i \neq a_j$, if outcome $\mathtt{da}$ is a possibility for a_i then it cannot also be a possibility for a_j – i.e., if $\mathtt{da} \in \mathcal{O}(a_i)$ then $\mathtt{da} \notin \mathcal{O}(a_j)$. We denote $\mu_a(\mathtt{da}_i)$ the (rational) probability that nature takes outcome $\mathtt{da}_i$, and for all a we require $\sum_{\mathtt{da}_i \in \mathcal{O}(a)} \mu_a(\mathtt{da}_i) = 1$. The outcome that nature chooses is observable, and its effect is given in terms of

[1] PLTL is called the "propositional linear temporal logic of the past" and \$FLTL is called the "future linear temporal logic with a dollar".

two lists of propositions called the add list, $add(\texttt{da})$, and delete list, $delete(\texttt{da})$. If a proposition is in the add list of da, then it cannot be in the delete list and *vice versa*. If outcome da with $add(\texttt{da}) := [p_1, .., p_n]$ and $delete(\texttt{da}) := [p'_1, .., p'_m]$ is executed at state s, then the resultant state is $(s \cup add(\texttt{da}))\backslash delete(\texttt{da})$ – i.e., propositions from $add(\texttt{da})$ are added to s, and those from $delete(\texttt{da})$ are removed from s.

3 Non-Markovian Rewards

A decision process with non-Markovian rewards is identical to an MDP except that the domain of the reward function is $\mathcal{S}^*$, the set of finite sequences of states over $\mathcal{S}$. We adopt the notation Γ for a member of $\mathcal{S}^*$, and where i is a natural number, Γ_i is the state at index i in Γ, and $\Gamma(i)$ is the prefix $\langle \Gamma_0, \ldots, \Gamma_i \rangle \in \mathcal{S}^*$ of Γ. In an NMRDP, at Γ_i the agent receives the reward associated with the history $\Gamma(i)$. There are two important observations to be made about NMRDPs. First, there are NMRDP models for which all optimal policies depend on the history of states traversed by the agent. Consequently, efficient MDP solution methods cannot be applied directly to optimal planning in NMRDPs. Fortunately, every non-Markovian model can be translated into an equivalent finite state MDP that is amenable to state-of-the-art MDP solution procedures. In theory then, we can solve an NMRDP by first translating it into an equivalent MDP, and then solve the latter. All proposed solution methods proceed in that vein. First appearing in (Bacchus et al., 1996), the criteria for MDP-NMRDP equivalence is given in Def 1.

Definition 1. *MPD-NMRDP Equivalence*

An MDP $\langle \mathcal{S}', \mathcal{A}, Pr', \mathsf{R}' \rangle$ is equivalent to an NMRDP $\langle \mathcal{S}, \mathcal{A}, Pr, \mathsf{R} \rangle$, if there exists mappings, surjective function $\tau : \mathcal{S}' \to \mathcal{S}$, and injective function $\sigma : \mathcal{S} \to \mathcal{S}'$, such that:

1. *$\forall s \in \mathcal{S}$ we have $\tau(\sigma(s)) = s$*
2. *For all $s_1, s_2 \in \mathcal{S}$, $a \in \mathcal{A}$, and $s'_1 \in \mathcal{S}'$, if $Pr(s_1, a, s_2) = p$ is non-zero and $\tau(s'_1) = s_1$, then there exists a $s'_2 \in \mathcal{S}'$ such that $\tau(s'_2) = s_2$ and $Pr'(s'_1, a, s'_2) = p$*
3. *For sequences of states Γ' and Γ which can occur with non-zero probability in the MDP and NMRDP respectively, where $\tau(\Gamma'_i) = \Gamma_i$ and $\sigma(\Gamma_0) = \Gamma'_0$, we have $\mathsf{R}(\Gamma(i)) = \mathsf{R}'(\Gamma'_i)$.*

In Def 1, $\mathsf{R}' : \mathcal{S}' \to \mathbb{R}$ is the stationary reward function of the equivalent MDP, and $\mathsf{R} : \mathcal{S}^* \to \mathbb{R}$ is the non-stationary reward function of the NMRDP. Function σ maps each NMRDP state s to an MDP state s' that models the case where $s = \Gamma_0$ – i.e. is the first state, or starting state of the NMRDP. Conditions 2 and 3 say that the MDP and NMRDP models are equivalent wrt their dynamics and reward structure.

Although one can painstakingly hand engineer equivalent MDPs, we are interested in procedures which automate this task. Three proposals have been

made, all of them suffering in one way or another from the fact that an equivalent MDP can have exponentially more states than the corresponding NMRDP. In each case that problem is tackled differently. Two approaches adopt a propositional linear temporal logic of the past (PLTL) to describe non-Markovian rewards. In the first (see Bacchus et al., 1996), a preprocessing step translates the entire NMRDP into an equivalent MDP. PLTL is motivated in this case because it enables one to easily produce an equivalent MDP with the minimal number of states. The second proposal (see Bacchus et al., 1997) describes how compact factored representations of the reward, transition, and value functions of the equivalent MDP can be used to achieve scalable procedures for solving structured problems. A third and later approach uses the forward looking propositional linear temporal logic $FLTL to describe NMRDP rewards. That choice was motivated by the success of online MDP solutions techniques, such as LAO* (Hansen and Zilberstein, 2001), which in practice find very good policies without evaluating all the states of a model. $FLTL logic has a number of advantages in that online setting. These are discussed in Thiébaux et al. (2006), who give a detailed empirical analysis of all the approaches just discussed.

4 PLTL Reward Logic

The first logic proposed for describing rewarding behaviours in NMRDP was PLTL. That is a propositional logic with 4 additional temporal operators: $\ominus$, unary prefix operator intuitively saying "in the last state", $\boxminus$ unary prefix"always in the past", $\diamondminus$ unary prefix "sometime in the past", and **S** binary infix operator "since". A formula f describes a set of finite sequences of states B_f, where $B_f \equiv \{\Gamma(i) | \Gamma(i) \models f\}$. Where f, g are formulae and p is a proposition, the modelling relation $\models$ is defined as follows:

$$
\begin{array}{ll}
\Gamma(i) \models p & \text{iff } p \in \Gamma_i \\
\Gamma(i) \models \neg f & \text{iff } \Gamma(i) \not\models f \\
\Gamma(i) \models f \wedge g & \text{iff } \Gamma(i) \models f \ \ and \ \ \Gamma(i) \models g \\
\Gamma(i) \models \ominus f & \text{iff } i > 0 \ \ and \ \ \Gamma(i-1) \models f \\
\Gamma(i) \models f \,\mathsf{S}\, g & \text{iff } \exists j \leq i, \Gamma(j) \models g \ \ and \ \ \forall k, j < k \leq i, \Gamma(k) \models f
\end{array}
$$

Operators $\boxminus$ and $\diamondminus$ are syntactic sugar: $\diamondminus f$ iff $\top \,\mathsf{S}\, f$, and $\boxminus f$ iff $\neg \diamondminus \neg f$. Using this logic the NMRDP model is given by: 1) a propositional model of the starting state and system dynamics, as discussed in Section 2, and 2) a PLTL reward model. The latter is a set of PLTL formulae each associated with a real-valued reward. Taking $\Gamma(k) \models f$ to have value 1 if true and 0 otherwise, the reward allocated at $\Gamma(k)$ is:

$$\mathsf{R}(\Gamma(k)) = \sum_{\langle f_i, r_i \rangle} (\Gamma(k) \models f_i).r_i$$

For i ranging over indices of elements in the set of rewarding formula. A key topic of our work is the expressive power of PLTL and related formalisms. In other words, the rewarding behaviours B_f that can be described using propositional logics from the decision-theoretic planning literature.

5 $FLTL Reward Logic

$FLTL is a future linear temporal logic (Emerson, 1990) augmented with a special propositional constant '$'. The latter can be thought of as a 0-ary past operator whose interpretation is determined according to where it appears in a formula. This logic was the subject of a detailed empirical study by Thiébaux et al. (2006) and was examined from a theoretical perspective by Slaney (2005). To define $FLTL, we start with a much more general language, denoted $\mathcal{F}\$$. That language has literals given by:

$$\ell ::= \mathcal{P}|\neg\mathcal{P}|\top|\bot|\$$$

where $\top$ and $\bot$ stand for *true* and *false* respectively, and $\mathcal{P}$ is a proposition. Notice that $\neg\$$ is not a literal in this logic. A formula in $\mathcal{F}\$$ is generated by the following grammar:

$$\mathcal{F} ::= \ell|\mathcal{F} \wedge \mathcal{F}|\mathcal{F} \vee \mathcal{F}|\bigcirc\mathcal{F}|\mathcal{F} \,\mathsf{U}\, \mathcal{F}|\Box\mathcal{F}$$

The $\Box$ operator is syntactic sugar, where $\Box f$ iff $f \,\mathsf{U}\, \bot$, thus we do not consider its semantics below. The semantics requires a behavioral index B, which is a set of sequences of states. For finite $i \in \mathbb{N}$ we have that formula f models a sequence of states $\Gamma(i)$ provided $(\Gamma, i) \models_B f$. The consequence relation $\models_B$ is defined as follows:

$$\begin{array}{lll}
(\Gamma,i) \models_B \$ & \text{iff } \Gamma(i) \in B \\
(\Gamma,i) \models_B \top & \\
(\Gamma,i) \not\models_B \bot & \\
(\Gamma,i) \models_B p & \text{iff } p \in \Gamma_i \text{ – Here, p is a proposition} \\
(\Gamma,i) \models_B \neg p & \text{iff } (\Gamma,i) \not\models_B p \\
(\Gamma,i) \models_B f \wedge g & \text{iff } (\Gamma,i) \models_B g \text{ and } (\Gamma,i) \models_B f \\
(\Gamma,i) \models_B f \vee g & \text{iff } (\Gamma,i) \models_B g \text{ or } (\Gamma,i) \models_B f \\
(\Gamma,i) \models_B \bigcirc f & \text{iff } (\Gamma,i+1) \models_B f \\
(\Gamma,i) \models_B f \,\mathsf{U}\, g & \text{iff } \forall k \geq i \text{ if } \forall j, i \leq j \leq k \\
 & \quad (\Gamma,j) \not\models_B g \text{ then } (\Gamma,k) \models_B f
\end{array}$$

$\mathcal{F}\$$ semantics only differ in the first line—i.e. the interpretation of the special reward proposition $—from those of classical *forward* linear temporal logic with the *weak until*. To see how $\mathcal{F}\$$ formulae can delineate rewarding behaviours we must consider the behavioural index B. For formula f we intend a behavioural index B_f defined in terms of $\models_B$. The behaviour B_f intended by f is the smallest set of sequences of states that make f a theorem. Formally, B_f satisfies the following equivalence:

$$B_f \equiv \bigcap\{B \mid \models_B f\} \tag{1}$$

The structure of an $\mathcal{F}\$$ specified NMRDP is the same as in the PLTL case, except the *reward formulae*—i.e. temporal logic formulae describing the non-

Markovian rewards—are written in $\mathcal{F}\$$. For pair $\langle f, r\rangle$, reward r is allocated to history $\Gamma(i)$ iff $\Gamma(i) \in B_f$.

$\mathcal{F}\$$ was developed for describing NMRDPs for online search-based solution procedures. Those procedures determine whether a state sequence is a rewarding behaviour as successive states in that sequence are discovered during search. For that purpose the technique of *formula progression* from model checking is used. Formula progression is a syntactic operation. Given f, progression produces a formula f' that is true at step $i+1$ iff f is true at step i. Treating $\mathcal{F}\$$, let f be a formula that describes the rewarding behavior at Γ_i. The progression operator yields f', a formula that describes a rewarding behavior available from state Γ_{i+1} onwards. Taking $\$\text{prg}_\$$ as defined in Alg 1, we have $f' = \$\text{prg}_\$(\Gamma_i, f)$.

Algorithm 1 $\mathcal{F}\$$ Progression

$\text{Rew}(\Gamma_i, f)$	=	$\top$ iff $\text{prg}_\$(\bot, \Gamma_i, f) = \bot$
$\$\text{prg}_\$(\Gamma_i, f)$	=	$\text{prg}_\$(\text{Rew}(\Gamma_i, f), \Gamma_i, f)$
$\text{prg}_\$(\top, \Gamma_i, \$)$	=	$\top$
$\text{prg}_\$(\bot, \Gamma_i, \$)$	=	$\bot$
$\text{prg}_\$(b, \Gamma_i, \top)$	=	$\top$
$\text{prg}_\$(b, \Gamma_i, \bot)$	=	$\bot$
$\text{prg}_\$(b, \Gamma_i, p)$	=	$\top$ iff $p \in s$ and $\bot$ otherwise
$\text{prg}_\$(b, \Gamma_i, \neg p)$	=	$\top$ iff $p \notin s$ and $\bot$ otherwise
$\text{prg}_\$(b, \Gamma_i, f_1 \wedge f_2)$	=	$\text{prg}_\$(b, \Gamma_i, f_1) \wedge \text{prg}_\$(b, \Gamma_i, f_2)$
$\text{prg}_\$(b, \Gamma_i, f_1 \vee f_2)$	=	$\text{prg}_\$(b, \Gamma_i, f_1) \vee \text{prg}_\$(b, \Gamma_i, f_2)$
$\text{prg}_\$(b, \Gamma_i, \bigcirc f)$	=	f
$\text{prg}_\$(b, \Gamma_i, f_1 \mathsf{U} f_2)$	=	$\text{prg}_\$(b, \Gamma_i, f_2) \vee(\text{prg}_\$(b, \Gamma_i, f_1) \wedge f_1 \mathsf{U} f_2)$

Above, the function $\text{Rew}(\Gamma_i, f)$ evaluates to *true* if and only if the formula f would be false at state Γ_i taking $\$ = \bot$. The first argument to $\text{prg}_\$$ is a Boolean giving us the truth value of $\$$ at state Γ_i. Alg 1 is therefore only admissible where "$\text{Rew}(\Gamma_i, f) = \top$ iff $\Gamma(i) \in B_f$" holds. The function Rew is employed in practice to determine whether a state sequence $\Gamma(i)$ implies a reward at Γ_i or not. In detail, take f to be a reward formula and let:

$$f^i := \$\text{prg}_\$(\Gamma_{i-1}..., \$\text{prg}_\$(\Gamma_1, \$\text{prg}_\$(\Gamma_0, f))).$$

In the NMRDP, the numerical reward associated with f is achieved at Γ_i iff $\text{Rew}(\Gamma_i, f^i) = \top$.

In general Alg 1 cannot be used as a tool for deciding if $\Gamma \in B_f$ according to Eq 1. For example, in the case of formula "$\$ \vee \bigcirc\$$" the above would allocate reward deterministically, and arguably incorrectly. In the case of "$\bigcirc p \rightarrow \$$", Alg 1 is simply wrong. In practice we only concern ourselves with the fragment of $\mathcal{F}\$$ where Alg 1 and Eq 1 make sense and are in correspondence. From hereon we shall restrict our attention to the fragment used in practice, given in Def 2.

Definition 2. *$FLTL*

$FLTL is the fragment of $\mathcal{F}$\$ that is generated by the following grammar.

$$\mathcal{F} ::= \$ | \bigcirc \mathcal{F} | M \vee \mathcal{F} | \mathcal{F} \mathsf{U} M | \mathcal{F} \wedge \mathcal{F}$$

Here, M is a modality free formula with no occurrences of the \$ symbol.

The grammar of Def 2 provides us with reward formulae that satisfy an intuition of "no funny business" (Slaney, 2005). Essentially, $FLTL makes sense wrt Alg 1 and Eq 1.

6 (Noncounting) Regular Languages and LTL Models

We now compare the expressive power of the logics for describing rewarding behaviours. We begin by reviewing a few prerequisite concepts from language theory. A language of finite strings is a set of finite-length words drawn from a finite set Σ, called the alphabet. Classic examples of formal languages from computer science include regular languages and context free languages. To describe a language, one will invoke an abstract structure that delineates a set of words. Regular languages can be specified using regular expressions.

Definition 3. *Where α is an element in the alphabet, $\alpha \in \Sigma$, regular expressions satisfy the following grammar.*

$$\text{regxp} ::= \alpha | \text{regxp} \cup \text{regxp} | \text{regxp} \circ \text{regxp} | \text{regxp}*$$

We write regxp^+ *to abbreviate* $\text{regxp} \circ \text{regxp}^*$. *To interpret an expression, $\cup$ is union, $\circ$ is concatenation, and $*$ is Kleene closure. We write ϵ for the empty string.*[2]

An important subfamily of the regular languages are the *noncounting* languages, also called *star-free regular* languages.

Definition 4. *A regular language $\mathcal{X}$ is noncounting iff there exists a finite n so that forall $x, y, z \in \Sigma^*$, $x \circ y^n \circ z \in \mathcal{X}$ iff $x \circ y^{n+1} \circ z \in \mathcal{X}$. Here, Σ^* is the Kleene closure of the alphabet and y^n is shorthand for n concatenated occurrences of y.*

Every regular language that can be specified as a star-free regular expression is noncounting. A short proof of this is given by Naughton and Papert (1971). Moreover, noncounting regular languages are a proper subfamily of the regular languages. That star-free regular expressions are able to describe any noncounting language requires a more substantial investment of space to prove.

Relating language in the above sense to execution histories in a decision process, we have already seen that a state s is the set of propositions that are *true* when the process is in s. In other words, where $\mathcal{P}$ is the finite set of state characterising propositions we have $\mathcal{S} \subseteq 2^{\mathcal{P}}$. In discussing behaviour as language we take as our finite alphabet the powerset of propositions $\Sigma \equiv 2^{\mathcal{P}}$.

[2] Given the available space, we must assume reader familiarity with the basic concepts and intuitions of language theory in computer science.

6.1 PLTL Behaviours are Noncounting Regular

We can now characterise the relative expressiveness of PLTL and $FLTL in terms of these concepts. We begin by reviewing the case of PLTL. That behaviours are regular in this case follows from the fact that there is an equivalent MDP (i.e. *finite stochastic automaton*) associated with every NMRDP with a finite reward specification. That counting languages cannot be described using PLTL can be shown by structural induction as follows.

Theorem 1. *PLTL Behaviours are Noncounting*

Proof. *That behaviours are noncounting for the PLTL fragment confined to operators $\wedge, \vee, \neg$ and $\ominus$ is straightforward. Hence our proof shall focus on the binary infix operator $\boldsymbol{S}$. Suppose $f, g \in PLTL$ are formulae that denote noncounting behaviours. We can augment any model Γ with propositions ("temporal variables") p_f and p_g to obtain a new model $\Gamma[p_f, p_g]$ satisfying: $\forall i\ \Gamma[p_f, p_g](i) \models p_f$ iff $\Gamma(i) \models f$, and $\Gamma[p_f, p_g](i) \models p_g$ iff $\Gamma(i) \models g$. For any model Γ and its augmented counterpart $\Gamma[p_f, p_g]$, we have that for any i, $\Gamma(i) \models f \boldsymbol{S} g$ iff $\Gamma[p_f, p_g](i) \models p_f \boldsymbol{S} p_g$. Now consider $\Sigma^*[p_f, p_g]$ obtained by augmenting all state sequences in Σ^* as above. Given f and g are noncounting, there must be star-free regular expressions e_f and e_g delineating languages $\mathcal{X}_{e_f}$ and $\mathcal{X}_{e_g}$ respectively, satisfying: $\Gamma(i) \models f$ iff $\Gamma(i) \in \mathcal{X}_{e_f}$, and $\Gamma(i) \models g$ iff $\Gamma(i) \in \mathcal{X}_{e_g}$. Therefore, $p_f \in \Gamma[p_f, p_g]_i$ iff $\Gamma(i) \in \mathcal{X}_{e_f}$, and $p_g \in \Gamma[p_f, p_g]_i$ iff $\Gamma(i) \in \mathcal{X}_{e_g}$. Let $\mathcal{X}(e_f, e_g)$ be any regular expression, and $\mathcal{X}(p_f, p_g)$ the same expression with subexpressions e_f and e_g replaced with alphabet symbols for p_f and p_g respectively. Because p_f and p_g are redundant in $\Sigma^*[p_f, p_g]$, if $\mathcal{X}(p_f, p_g)$ is noncounting using $\Sigma^*[p_f, p_g]$ rather than Σ^* in Def 4, then $\mathcal{X}(e_f, e_g)$ is noncounting using Σ^*. Now, let $\mathcal{X}(p_f, p_g) := (\Sigma^*[p_f, p_g] \circ p_g) \circ p_f^*$. Observe that $\forall i\ \Gamma[p_f, p_g](i) \in \mathcal{X}(p_f, p_g)$ iff $\Gamma[p_f, p_g](i) \models p_f \boldsymbol{S} p_g$. Both p_f^* and $\Sigma^*[p_f, p_g]$ are obviously noncounting taking $\Sigma^*[p_f, p_g]$ as Σ^* in Def 4, otherwise the expression $\mathcal{X}(e_f, e_g)$ is star-free and therefore noncounting. Consequently $f \boldsymbol{S} g$ is noncounting, so the theorem holds.*

We have just proved that PLTL can only describe behaviours that correspond to languages that are noncounting. In wrapping up this section it is worth noting that Thm 1 is stated by Lichtenstein et al. (1985), however no proof was given. Otherwise, in the literature Thm 1 is stated as a consequence of a symmetry with another LTL, and not directly and from first principles as we have shown.

6.2 $FLTL Behaviours

Dealing exclusively with $FLTL, we have that questions regarding the expressive power of the logic have yet to be answered. We know, because it is first-order definable (see Slaney, 2005), that it can only describe behaviors that correspond to regular languages. Here, we show that behaviours expressible in $FLTL are noncounting.

Theorem 2. *$FLTL Behaviours are Noncounting Regular*

Proof. Clearly none of formulae \$, $\neg p$, p, $\bot$ *or* $\top$ *specifies a counting language. Non-dollar propositional formulae* f *have* $B_f \equiv \Sigma^+$ *or else* $B_f \equiv \emptyset$. *Formula* \$ *specifies the set of state sequences of length* 1. *For the formula* $\bigcirc f$, *we have:* $B_{\bigcirc f} \equiv \{\Sigma \circ \Gamma | \Gamma \in B_f\}$. *Thus, if* B_f *is noncounting so is* $B_{\bigcirc f}$. *Taking noncounting* f_1 *and* f_2, *for* $f_1 \wedge f_2$ *we have* $B_{f_1 \wedge f_2} \equiv B_{f_1} \cup B_{f_2}$ *which is not counting. Also, if* $\Gamma \in B_{f_1 \vee f_2}$ *then* $\Gamma \in B_{f_1}$ *or else* $\Gamma \in B_{f_2}$ – *i.e., by definition either* f_1 *or* f_2 *is free of* \$ *symbols and modalities. Thus, we cannot get a counting language from disjunction. In the case* $f \mathsf{U} m$ *where* m *is propositional, we have* $B_{f \mathsf{U} m} \equiv (L')^* \circ (B_f \cap (L')^*)$ *where* $L' = \{s | s \not\models_B m\}$ *for any* B. *Thus* U *does not give us a counting language.*

7 \$*FLTL: A More Expressive LTL for Planning

We are thus motivated to enhance \$FLTL to obtain a logic equal in expressive power to regular expressions. We only enhance the forward looking language because our primary interest lies in online search-based planning procedures, a setting that is ill-suited to PLTL approaches. To increase the expressive power of \$FLTL we design formulae in which multiple occurrences of distinguishable reward propositions occur. Moreover, we shall do that without increasing the computational cost of formula progression, and therefore the computational cost of using our new logic to describe rewards in practice. We call the resulting logic \$*$FLTL$. To describe it, we begin by considering a future looking linear temporal logic $\mathcal{F}\$^*$ given by the grammar:

$$\mathcal{F} ::= M | \$_i | \top | \bot | \mathcal{F} \rightarrow \mathcal{F} | \neg \mathcal{F} | \bigcirc \mathcal{F} | \mathcal{F} \mathsf{U} \mathcal{F}$$

This differs from \$FLTL in two key ways: 1) there are now many \$ symbols, each indexed by an element in the natural numbers, and 2) a negation normal formula in $\mathcal{F}\$^*$ can have subformulae of the form $\neg\$_i$. These minor differences are sufficient for us to create a much more powerful logic with only minor modifications. For $\mathcal{F}\$^*$ the modelling relation is indexed by a sequence of behaviours, written $\mathbb{B}$, rather than by a single behavior/history. The semantics differ notably from \$FLTL only in the interpretation of \$ symbols,

$$(\Gamma, i) \models_{\mathbb{B}} \$_j \text{ iff } \Gamma(i) \in \mathbb{B}(j)$$

Using $\mathcal{F}\$^*$ we can describe the union of behaviours as $\$_1 \vee \$_2$ (e.g. $(\Gamma, i) \models_{\mathbb{B}} \$_1 \vee \$_2$ then $\Gamma(i) \in \mathbb{B}(1) \cup \mathbb{B}(2)$), their intersection as $\$_1 \wedge \$_2$ and the complement of a behaviour as $\neg\$_1$. We write about behavioural equivalences $\models_{\mathbb{B}} \Box(\$_1 \leftrightarrow \$_2)$, which says for all Γ and i, $\Gamma(i) \in \mathbb{B}(1)$ iff $\Gamma(i) \in \mathbb{B}(2)$, or equivalently $\mathbb{B}(1) \equiv \mathbb{B}(2)$.[3] The universal quantification over Γ comes from the original statement having the form $\models_{\mathbb{B}} \phi$ for some $\phi \in \mathcal{F}\$^*$. Universal quantification over i comes from the $\Box$ modality. In a similar way we write about sub/super-set relationships between behaviours via $\models_{\mathbb{B}} \Box(\$_1 \rightarrow \$_2)$, which says $\mathbb{B}(1) \subseteq \mathbb{B}(2)$.

[3] Recall, unary operator $\Box$ is syntactic sugar for $f \mathsf{U} \bot$.

We now consider the behaviors we intend by formula f. Let $\mathbb{B}^*_f$ be the set of sequences of behaviours which make f a theorem:

$$\mathbb{B}^*_f \equiv \{\mathbb{B} | \models_{\mathbb{B}} f\}$$

To interpret the rewarding behaviour described by f, here we restrict ourselves to the situation where we choose the behaviour for each reward-proposition according to the following inductive scheme.

$$\begin{array}{l} \mathbb{B}_f(0) = \bigcap_{\mathbb{B}\in\mathbb{B}^*_f} \mathbb{B}(0) \\ \mathbb{B}_f(i) \ = \bigcap_{\mathbb{B}\in\mathbb{B}^*_f, \mathbb{B}(0)=\mathbb{B}_f(0),\ldots\mathbb{B}(i-1)=\mathbb{B}_f(i-1)} \mathbb{B}(i) \end{array} \tag{2}$$

In other words, the behaviour $\$_0$ is minimal, that of $\$_1$ is minimal given $\mathbb{B}_f(0)$, and generally the behaviour described by a $\$_i$ is determined after those of $\$_j$ for $j < i$. A fragment of $\mathcal{F}\$^*$ that satisfies Eq 2, which we call $\$^*FLTL$, is given by:

$$\begin{array}{ll} \mathcal{F} ::= M \rightarrow \$_i & \text{letter-rule} \\ \ |\mathcal{F}_j \wedge \mathcal{F}_k \wedge \Box((\$_j \wedge \$_k) \rightarrow \$_i) \ \{j < k < i\} & \text{intersection} \\ \ |\mathcal{F}_j \wedge \mathcal{F}_k \wedge \Box((\$_j \vee \$_k) \rightarrow \ \$_i) \ \{j < k < i\} & \text{union} \\ \ |\mathcal{F}_j \wedge \Box(\$_j \vee \$_i) \ \{j < i\} & \text{complement} \\ \ |\mathcal{F}_i \wedge \mathcal{F}_{i+1} \wedge \ldots \wedge \mathcal{F}_n \wedge \Box(\$_n \rightarrow (\bigcirc(\mathcal{F}_{n+1} \wedge \mathcal{F}_{n+2} \wedge \ldots \wedge \mathcal{F}_{n+m}))) & \text{concatenation} \\ \ |\mathcal{F}_i \wedge \mathcal{F}_{i+1} \wedge \ldots \wedge \mathcal{F}_n \wedge \Box(\$_n \rightarrow (\$_{n+1} \wedge \bigcirc(\mathcal{F}_i \wedge \mathcal{F}_{i+1} \wedge \ldots \wedge \mathcal{F}_n))) & \text{star} \end{array}$$

Above, M is a propositional formula. All $\mathcal{F}_i$ are formula that satisfy the following index criteria. If $\mathcal{F}_i$ is not an instance of the star-rule, then $\$_i$ is the rightmost occurrence of a reward proposition in $\mathcal{F}_i$. Otherwise, if $\mathcal{F}_i$ is a star-rule then $\$_i$ is the second reward-proposition appearing to the right of the $\Box$ operator (as it appears in the grammar). For example, $\mathcal{F}_3$ could be the formula "$p \rightarrow \$_1 \wedge q \rightarrow \$_2 \wedge \Box((\$_1 \wedge \$_2) \rightarrow \$_3)$", and $\mathcal{F}_2$ might be "$p \rightarrow \$_1 \wedge \Box(\$_1 \rightarrow (\$_2 \wedge \bigcirc(p \rightarrow \$_1)))$".

Theorem 3. *An $\$^*FLTL$ formula can express any regular language.*

Proof. We have access to all alphabet symbols because a propositional formula M can imply a reward proposition – i.e. $M \rightarrow \$_i$. We also have access to the union, intersection, complement, concatenation and Kleene closure via their respective labelled rules in the $\$^*FLTL$ grammar.

To specify an NMRDP reward function R using $\$^*FLTL$, for each rewarding behavior we give a reward formula f, again associate it with the numeric reward value, and additionally provide the index of the $\$_i$ symbol where $\mathbb{B}_f(i)$ is the rewarding behavior.

We are left to describe a linear time formula progression procedure for \$FLTL. If f is a *letter-rule* formula then Alg 1 is sufficient – i.e. treating $\$_i$ as \$. For the other formula types we require a few building blocks. First, we require an $\$^*FLTL$ analogue of function Rew from Alg 1. In other words, a function that

tells us when $\Gamma(k) \in \mathbb{B}_f(i)$ for each $\$_i \in f$. Below we adopt the notation $f\lceil \$_i/C \rfloor$ to describe the formula f in which every occurrence of the symbol $\$_i$ is replaced with the expression C.[4] Let $\mathcal{F}$ be the set of all $\$^*FLTL$ formulae and consider the function $\text{Rew}^* : \{0,1\}^* \times \mathcal{S} \times \mathcal{F} \times \mathbb{N} \to \{0,1\}$ where:

$$\text{Rew}^*(\epsilon, s, f, i) = \begin{array}{l} \top \text{ if } \text{prg}_\$(\bot, s, f\lceil \$_i/\$ \text{ and } \$_j/\top \text{ for } j > i \rfloor) = \bot \\ \bot \text{ otherwise} \end{array}$$

in the case that the first argument is the empty string, and otherwise where it is defined,

$$\text{Rew}^*(\mathtt{B}, s, f, i) = \text{Rew}^*(\epsilon, s, f\lceil \$_j/\mathtt{B}_j \text{ for } j < i \rfloor, i).$$

Recall $\text{prg}_\$$ is described in Alg 1. Given a state s and formula f, we can obtain a Boolean string $\mathtt{B}$ whose i'th element, written $\mathtt{B}_i$, determines the truth of reward-proposition $\$_i \in f$ at s using the following recursion:

$$\begin{array}{l} \mathtt{B}(0) = \text{Rew}^*(\epsilon, s, f, 0) \\ \mathtt{B}(i) \ = \mathtt{B}(i-1) \circ \text{Rew}^*(\mathtt{B}(i-1), s, f, i) \end{array}$$

If f is the reward formula at state s, Rew^* can be used to tell us the values of $\$_i$ symbols in f at s. We are left to describe a formula progression that takes f and $\mathtt{B}(i)$ (here, i is the largest index of a \$ term in f) producing f', a formula that describes rewards at Γ_{k+1} iff f is true at Γ_k. Taking $\text{prg}^*_\$$ as defined in Alg 2 and $s = \Gamma_k$, we have that $f' = \text{prg}^*_\$(\mathtt{B}(i), s, f)$. Because terms in a conjunction can be progressed independently and are limited to be of constant length, the structure of $\$^*FLTL$ means Alg 2, including the computation of $\mathtt{B}(i)$, can be implemented to run in linear time in the length of the input formula. In other words, $\$^*FLTL$ and \$FLTL progression have the same runtime complexity.

On a final note, we developed $\$^*FLTL$ in order to make the statement of Thm 3 straightforward. In practice it is a small matter to modify Alg 2 to also support \$FLTL formulae. Essentially, $\$^*FLTL$ sketches how to enhance \$FLTL so that reward engineers can keep their existing corpus of formulae, while adding much richer reward models that cannot be described using \$FLTL.

8 Conclusion and Related Work

There are well established formal relationships between monadic theories of linear ordering, regular languages and classical propositional linear temporal logic – see (Emerson, 1990) and (Wolper, 1983). In that tradition we have studied the relationship between regular languages, star-free (or noncounting) regular languages, PLTL and \$FLTL. The latter are logics that are used for describing rewarding behaviours in decision-theoretic planning with non-Markovian rewards.

[4] For multiple replacements, written $f\lceil \$_1/C_1\$_2/C_2 \rfloor$, it makes no difference whether replacement be taken as consecutive (replacing for index 1 and then 2) or simultaneous (replacing at the same time for all indices).

Algorithm 2 $\$^*FLTL$ Progression

$$
\begin{array}{ll}
\mathrm{prg}^*_\$(\mathtt{B}, s, \top) & = \top \\
\mathrm{prg}^*_\$(\mathtt{B}, s, \bot) & = \bot \\
\mathrm{prg}^*_\$(\mathtt{B}, s, \$_i) & = \top \text{ if 1 is the i'th bit in } \mathtt{B} \text{ otherwise, } \bot \\
\mathrm{prg}^*_\$(\mathtt{B}, s, p) & = \top \text{ if } p \in s \text{ otherwise } \bot \\
\mathrm{prg}^*_\$(\mathtt{B}, s, f \rightarrow g) & = \mathrm{prg}^*_\$(\mathtt{B}, s, f) \rightarrow \mathrm{prg}^*_\$(\mathtt{B}, s, g) \\
\mathrm{prg}^*_\$(\mathtt{B}, s, f \wedge g) & = \mathrm{prg}^*_\$(\mathtt{B}, s, f) \wedge \mathrm{prg}^*_\$(\mathtt{B}, s, g) \\
\mathrm{prg}^*_\$(\mathtt{B}, s, \Box(\$_j \vee \$_k)) & = \mathrm{prg}^*_\$(\mathtt{B}, s, \$_j \vee \$_k) \wedge \Box(\$_j \vee \$_k) \\
\mathrm{prg}^*_\$(\mathtt{B}, s, \Box((\$_i \vee \$_j) \rightarrow \$_k)) & = \mathrm{prg}^*_\$(\mathtt{B}, s, (\$_i \vee \$_j) \rightarrow \$_k) \wedge \\
 & \quad \Box((\$_i \vee \$_j) \rightarrow \$_k)) \\
\mathrm{prg}^*_\$(\mathtt{B}, s, \Box((\$_i \wedge \$_j) \rightarrow \$_k)) & = \mathrm{prg}^*_\$(\mathtt{B}, s, (\$_i \wedge \$_j) \rightarrow \$_k) \wedge \\
 & \quad \Box((\$_i \wedge \$_j) \rightarrow \$_k)) \\
\mathrm{prg}^*_\$(\mathtt{B}, s, \Box(\$_n \rightarrow (\$_{n+1} \wedge \bigcirc(f_i \wedge \ldots \wedge f_n)))) = & \\
\quad \begin{cases} f_i \wedge \ldots \wedge f_n \wedge \Box(\$_n \rightarrow (\$_{n+1} \wedge \bigcirc(f_i \wedge \ldots \wedge f_n))) & \text{if } \mathtt{B}_n = 1 \\ \Box(\$_n \rightarrow (\$_{n+1} \wedge \bigcirc(f_i \wedge \ldots \wedge f_n))) & \text{otherwise} \end{cases} & \\
\mathrm{prg}^*_\$(\mathtt{B}, s, \Box(\$_n \rightarrow (\bigcirc(f_{n+1} \wedge \ldots \wedge f_{n+m}))) = & \\
\quad \begin{cases} f_{n+1} \wedge \ldots \wedge f_{n+m} \wedge \Box(\$_n \rightarrow (\bigcirc(f_{n+1} \wedge \ldots \wedge f_{n+m}))) & \text{if } \mathtt{B}_n = 1 \\ \Box(\$_n \rightarrow (\bigcirc(f_{n+1} \wedge \ldots \wedge f_{n+m}))) & \text{otherwise} \end{cases} &
\end{array}
$$

We have discussed the expressive power of both \$FLTL and PLTL. We provided a proof that PLTL can only describe behaviours that correspond to star-free regular languages. We showed that the same is true for \$FLTL. Finding both \$FLTL and PLTL wanting of expressive power, we have developed a forward looking logic $\$^*FLTL$ which is able to describe behaviours that correspond to regular languages. The formulae we used to define $\$^*FLTL$ will give reward engineers much more flexibility in compactly expressing rich reward signals in decision-theoretic planning.

In closing, we note that $\$^*FLTL$ is by no means the first linear temporal logic with the power to describe any regular language. In particular Kesten and Pnueli (1995) and Emerson (1990) discuss various Quantified Propositional Temporal Logics (QPTL) and Wolper (1983) gives an Extended Temporal Logic (ETL). The key advantages of $\$^*FLTL$ are: 1) that model checking—i.e. as is implemented by formula progression in our setting—is relatively cheap, and 2) $\$^*FLTL$ is based closely on a logic for describing non-Markovian rewards in decision-theoretic planning.

Acknowledgments. NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

Bibliography

Bacchus, F., Boutilier, C., Grove, A.: Rewarding behaviors. In: Proc. American National Conference on Artificial Intelligence (AAAI). pp. 1160–1167 (1996)

Bacchus, F., Boutilier, C., Grove, A.: Structured solution methods for non-Markovian decision processes. In: Proc. American National Conference on Artificial Intelligence (AAAI). pp. 112–117 (1997)

Boutilier, C., Dean, T., Hanks, S.: Decision-theoretic planning: Structural assumptions and computational leverage. In: Journal of Artificial Intelligence Research. vol. 11, pp. 1–94 (1999)

Boutilier, C., Dearden, R.: Using abstractions for decision-theoretic planning with time constraints. In: AAAI. pp. 1016–1022 (1994)

Emerson, E.A.: Temporal and modal logic. In: Handbook of Theoretical Computer Science, vol. B, pp. 997–1072. Elsevier and MIT Press (1990)

Hansen, E., Zilberstein, S.: LAO*: A heuristic search algorithm that finds solutions with loops. Artificial Intelligence 129, 35–62 (2001)

Kesten, Y., Pnueli, A.: A complete proof systems for QPTL. In: Logic in Computer Science. pp. 2–12 (1995)

Lichtenstein, O., Pnueli, A., Zuck, L.: The glory of the past. In: Proc. Conference on Logics of Programs. pp. 196–218. LNCS, volume 193 (1985)

Naughton, R.M., Papert, S.: Counter-Free Automata. M.I.T. Press, Cambridge, Mass. (1971)

Slaney, J.: Semi-positive LTL with an uninterpreted past operator. Logic Journal of the IGPL 13, 211–229 (2005)

Thiébaux, S., Gretton, C., Slaney, J., Price, D., Kabanza, F.: Decision-theoretic planning with non-markovian rewards. J. Artif. Intell. Res.(JAIR) 25, 17–74 (2006)

Wolper, P.: Temporal logic can be more expressive. Information and Control 56(1/2), 72–99 (1983)