Partially Observable Markov Decision Processes with Behavioral Norms
(Extended abstract)

Matthias Nickles[1] and Achim Rettinger[2]

[1] Department of Computer Science, University of Bath
Bath, BA2 7AY, United Kingdom
m.l.nickles@cs.bath.ac.uk,
[2] Department of Computer Science, Technical University of Munich,
D-85748 Garching bei München, Germany
achim.rettinger@cs.tum.edu

Abstract. This extended abstract discusses various approaches to the constraining of Partially Observable Markov Decision Processes (POMDPs) using social norms and logical assertions in a dynamic logic framework. Whereas the exploitation of synergies among formal logic on the one hand and stochastic approaches and machine learning on the other is gaining significantly increasing interest since several years, most of the respective approaches fall into the category of relational learning in the widest sense, including inductive (stochastic) logic programming. In contrast, the use of formal knowledge (including knowledge about social norms) for the provision of hard constraints and prior knowledge for some stochastic learning or modeling task is much less frequently approached. Although we do not propose directly implementable technical solutions, it is hoped that this work is a useful contribution to a discussion about the usefulness and feasibility of approaches from norm research and formal logic in the context of stochastic behavioral models, and vice versa.
Keywords: Norms, Partially Observable Markov Decision Processes, Deontic Logic, Propositional Dynamic Logic

1 Introduction

This extended abstract discusses various approaches to the constraining of *Partially Observable Markov Decision Processes* (POMDPs) using social norms and logical assertions in a dynamic logic framework. Whereas the exploitation of synergies among formal logic on the one hand and stochastic approaches and machine learning on the other is gaining significantly increasing interest since several years, most of the respective approaches fall into the category of *relational learning* in the widest sense [12], including inductive (stochastic) logic programming. In contrast, the use of formal hard constraints and prior knowledge for other stochastic modeling or machine learning tasks is relatively seldom approached ("hard" constraint in the sense that the constraint cannot be

Dagstuhl Seminar Proceedings 09121
Normative Multi-Agent Systems
http://drops.dagstuhl.de/opus/volltexte/2009/1913

overwritten, ignored or weakened). Among the existing approaches which allow for the specification of hard constraints of stochastic tasks are [5, 11], and, closer to our work, various extensions of Golog, such as [7, 6]. [7, 6] allow for the logic-based partial specification of a program and the automatic and optimal completion of this program, which is viewed as a Markov Decision Process (respectively a POMDP in case of [7]). In contrast to these approaches, we propose to take an ordinary POMDP (which could be manually created or automatically learned) and a set of "ordinary" modal logic formulas, and use the latter in order to modify the given POMDP and/or a standard algorithm for solving this POMDP (i.e., the search for an optimal behavior policy) so that certain action sequences become impossible or less probable (because they would violate a norm), or obligatory. Besides this, our approach uses a variant of dynamic logic, whereas Golog is based on the situation calculus. We find dynamic logic more useful for dealing with complex actions (patterns and compositions of elementary actions) than the situation calculus, and specifically for the specification of norms about complex actions.
We believe that the normative constraining of a stochastic model of an agent's environment would be useful for various potential applications. Agents in a multiagent system are potentially subject to social norms (respectively, sanctions in case of norm violating behavior) and need to take these norms into consideration when they plan their behavior. If the environment is noisy and/or only partially accessible to the agent's perception, or - more generally - if the agent is uncertain about the state of its environment, it needs to maintain a stochastic model of this environment and act in dependency from uncertain beliefs - including the compliance or intentional noncompliance with any of the norms. While it would be possible for the agent to consult its "norm base" (or to query the normative system in some way) at each step, it appears to be much more efficient to embed the knowledge about norms directly into the stochastic decision model of the agent, in terms of expected positive or negative rewards in case of norm obedience or failure to do so, respectively.
Although we do not propose directly implementable technical solutions in this paper, it is hoped that this work is a useful contribution to a discussion about the usefulness and feasibility of approaches from norm research and formal logic in the context of stochastic uncertainty modeling, and vice versa.

The remainder of this work is organized as follows: the next section describes the deontic logic we use to represent behavioral norms. Section 3 provides a brief introduction of POMDPs. Section 4 outlines and discusses various possibilities for the logical constraining of POMDPs using our formal framework. Section 5 concludes.

2 Representing Norms Using Dynamic Logic

We use *Propositional Deontic Logic* (PD_eL) [1] to represent the norms [10] which the agent is subject to as well as the agent's knowledge. PD_eL is a variant of

Propositional Dynamic Logic (PDL) [14, 2], and has as such properties which are very handy in our context: it is not only a well-researched language with a sound axiom system, but its Kripke-style semantics is also relatively close to the representation of Markov processes we will use later. More precisely, our semantics of PD_eL uses Kripke structures where the meaning of a certain construct is defined in terms of the current state and actions which define transitions to one ore more other states. We will use this later to constrain a Partially Observable Markov Decision Process (POMDP) [4, 8, 13]. Furthermore, PD_eL deals well with conflicting obligations and avoids several paradoxes known from other deontic logics [1].

And finally, it can be shown that certain description logics are syntactic variants of PDL [3], a fact which might be useful in order to represent PD_eL encodable norms and knowledge on the Semantic Web. However, we have not investigated this possibility in this work.

PD_eL is a normal modal logic K with additional axioms for actions. The only modal operator $[\alpha]$ has more or less the same meaning as in standard dynamic logic and corresponds to an action-annotated necessity multi-modality $\Box_{action}$. I.e., $[\alpha]\phi$ denotes a sufficient precondition for ϕ after action α has been performed. Obligation, permission and prohibition are derived from this modality together with a special proposition V which is used as a marker for undesirable states. Although this appears at a first glance as a kind of work-around compared to deontic logics with a dedicated deontic modality, it is actually an elegant solution which nicely reflects the agent's rationale for observing a norm and furthermore allows for a straightforward mapping of Kripke states to the reward-annotated (i.e., more or less desired) states of a POMDP. We are aware of the limitations arising from the fact that our logical framework (but of course not the POMDPs) only has neutral and undesirable states but cannot express positive rewards directly. That is, we can only model negative sanctions. Future versions might overcome this limitation.

We assume in this paper that there is a single agent which would be negatively sanctioned if it would not observe all given norms as far as possible (that is, under the provision that the respective desired states are reachable), and that all norms are equally preferred. However, there is no principled reason why these assumptions could not be dropped, at the price of a technically somewhat more complex model.

In the following, we present an abbreviated account of PD_eL; for full details please refer to [1]. First, we introduce the following sets:

- A non-empty set Act of action expressions. Although PD_eL is not identical with PDL, we make use of the PDL terminology and refer to the elements of Act as programs.
- A non-empty set Act_0 of atomic actions.
- A non-empty set Φ of formulas.

which are the smallest sets satisfying the following conditions (with $\alpha, \alpha_1, \alpha_2 \in Act$, $\phi, \phi_1, \phi_2 \in \Phi$):

1. $A_0 \subset Act$
2. $\emptyset \in Act$, $Any \in Act$ ($\emptyset$ stands for "failure" (an impossible action with no successor state), Any for "any actions" (some non-deterministically choosen atomic actions are performed simultaneously.))
3. $\alpha_1; \alpha_2 \in Act$ (sequential composition)
4. $\alpha_1 \cup \alpha_2 \in Act$ (choice. Perform either α_1 or α_2.)
5. $\alpha_1 \& \alpha_2 \in Act$ (joint action. Perform α_1 and α_2 concurrently.)
6. $\phi \rightarrow \alpha_1/\alpha_2 \in Act$ (conditional action. If ϕ holds in the current state, perform α_1, and α_2 otherwise.)
7. $\overline{\alpha} \in Act$ (negated action. Not action α)
8. $V \in \Phi$ (the special proposition which marks "unpleasant states")
9. $\phi_1 \vee \phi_2, \phi_1 \wedge \phi_2, \phi_1 \rightarrow \phi_2, \phi_1 \equiv \phi_2, \neg\phi \in \Phi$
10. $[\alpha]\phi, < \alpha > \phi \in \Phi$ (with $<>$ being the dual of $[]$, with $\sigma \models < \alpha > \phi \Leftrightarrow_{def} \sigma \models \neg[\alpha]\neg\phi$)

Norms can be specified using the following abbreviations:

Prohibition $\sigma \models F\alpha \Leftrightarrow_{def} \sigma \models [\alpha]V$ (we say that it is *forbidden* to do α)
Obligation $\sigma \models O\alpha \Leftrightarrow_{def} \sigma \models F\overline{\alpha}$ (we say that the agent is *obliged* to do α)
Permission $\sigma \models P\alpha \Leftrightarrow_{def} \sigma \models \neg F\alpha$ (we say that it is *permitted* to do α)

2.1 Semantics and Axioms of PD_eL

The Kripke-style semantics, based on the semantics of PDL, is outlined in Section 4.1.

Axioms (in addition to those of propositional logic):

$$[\alpha](\phi_1 \rightarrow \phi_2) \rightarrow ([\alpha]\phi_1 \rightarrow [\alpha]\phi_2) \tag{1}$$
$$[\alpha_1; \alpha_2]\phi \equiv [\alpha_1]([\alpha_2]\phi) \tag{2}$$
$$[\alpha_1 \cup \alpha_2]\phi \equiv [\alpha_1]\phi \wedge [\alpha_2]\alpha \tag{3}$$
$$[\alpha_1]\phi \vee [\alpha_2]\phi \rightarrow [\alpha_1 \& \alpha_2]\phi \tag{4}$$
$$[\phi_1 \rightarrow \alpha_1/\alpha_2]\phi_2 \equiv (\phi_1 \rightarrow [\alpha_1]\phi_2) \wedge (\neg\phi_1 \rightarrow [\alpha_2]\phi_2) \tag{5}$$
$$< \alpha > \phi \equiv \neg[\alpha]\neg\phi \tag{6}$$
$$[\overline{\alpha_1; \alpha_2}]\phi \equiv [\overline{\alpha_1}]\phi \wedge [\alpha_1][\overline{\alpha_2}]\phi \tag{7}$$
$$[\overline{\alpha_1}]\phi \vee [\overline{\alpha_2}]\phi \rightarrow [\overline{\alpha_1 \cup \alpha_2}]\phi \tag{8}$$
$$[\overline{\alpha_1 \& \alpha_2}]\phi \equiv [\overline{\alpha_1}]\phi \wedge [\overline{\alpha_2}]\phi \tag{9}$$
$$[\overline{\phi_1 \rightarrow \alpha_1/\alpha_2}]\phi_2 \equiv (\phi_1 \rightarrow [\overline{\alpha_1}]\phi_2) \wedge (\neg\phi_1 \rightarrow [\overline{\alpha_2}]\phi_2) \tag{10}$$
$$[\overline{\alpha}]\phi \equiv [\alpha]\phi \tag{11}$$
$$[\emptyset]\phi \tag{12}$$

3 Partially Observable Markov Decision Processes

Partially Observable Markov Decision Processes (POMDPs) model an agent's decision problems in some environment where the agent's perception is limited or noisy [4, 8, 13]. The primary goal of the agent is to find an optimal action policy, that is, to find a sequence of decisions regarding its behavior such that its reward is maximized.

Formally, a POMDP is a tuple $(S, A_0, T, R, O, \Omega)$, where

- S is a finite, non-empty set of world states, denoted in this paper as "states" or "Markovian states" (the latter in demarcation from the larger set of world states used in the Kripke structures),
- A_0 is the finite set of atomic actions,
- $T : S \times A_0 \to \Pi(S)$ is the state-transition function. For each state and each (atomic) agent action $a \in A_0$ it yields a probability distribution over states. $T(\sigma, a, \sigma')$ stands for the probability that the agent ends in state σ' given it starts in state σ and performs atomic action a.
- Ω is the set of all possible observations the agent can make in its environment.
- $R : S \times A_0 \to \mathbb{R}$ is the agent's reward function. It yields for each action and each state the immediate reward $R(s, \alpha)$ for taking this action.
- $O : S \times A_0 \to \Pi(\Omega)$ is the observation function, which gives for each action and each resulting state a probability distribution over possible observations. $O(\sigma', a, o)$ stands for the probability of making observation o after performing atomic action $a \in A_0$ and ending with this action in state σ'.

We use the same symbol A_0 for the set of atomic actions in PD_eL, since both sets are actually identical in our framework. S should correspond to a set of states (worlds) in the Kripke-structures (cf. the next section).

The next state and the reward depend only on the current state and the performed action. That is, POMDPs fulfil the Markov property.

A POMDP models an uncertain part of the agent's subjective and dynamic beliefs about a noisy environment in which the agent takes action. However, we do not make this explicit in our *logical* framework (which would require us to introduce inter alia a doxastic modality and a probability distribution over states, as in, e.g., [9]). Instead we will later update a given, inaccurate POMDP with certain knowledge from our PD_eL knowledge base (KB). The KB and the POMDP can then either co-exist, or only the POMDP is maintained.

Given a POMDP, the agent's tasks are i) to update its belief state in dependency from its previous belief state, the agent's current observation, and the agent's last action and ii) to generate optimal actions, depending on the belief state and expected rewards.

A belief state is a probability distribution over world states. It can be seen as a roundup of the agent's initial belief state updated by its past experiences. Because of this, it is not required to take into account the history of past actions and observations explicitly for decision making. However, there are infinitely many belief states.

Formally, a belief state is a function $b : S \times [0;1]$ and $b(s)$ is the probability that the agent is in state s, with $\sum_{s \in S} b(s) = 1$.

Computing a new belief state $b' = \tau(b, \alpha, o)$ given the old belief state b, an action α and an observation o (*state estimation*) is not very complicated. $\tau(b, \alpha, o)$ is called the *belief state transition function*.

$$b'(s') = Pr(s'|o, \alpha, b) \tag{13}$$

$$= \frac{Pr(o|s', \alpha, b) Pr(s'|a, b)}{Pr(o|\alpha, b)} \tag{14}$$

$$= \frac{Pr(o|s', \alpha) \sum_{s \in S} Pr(s'|\alpha, b, s) Pr(s|a, b)}{Pr(o|\alpha, b} \tag{15}$$

$$= \frac{O(s', \alpha, o) \sum_{s \in S} T(s, \alpha, s') b(s)}{Pr(o|\alpha, b)} \tag{16}$$

The belief states together with their updating function form a certain kind of *observable* Markov Decision Process, a so-called *continuous state space belief-MDP*. This insight is crucial for solving a POMDP, since it allows to formulate the solution of the POMDP (i.e., the optimal behavioral policy) as the solution of this kind of MDP.
The belief-MDP is defined as a tuple (B, A_0, τ, r), with:

- B being the set of belief states, as defined above,
- A_0 being the same action set as for the POMDP,
- τ being the belief state transition function, and
- $r : B \times A_0 \rightarrow \mathbb{R}$, the reward function of the belief states, with $r(b, a) = \sum_{s \in S} b(s) R(s, a)$ (R is the agent's reward function as defined for the POMDP, i.e., operating on actual world states instead of uncertain beliefs about such states).

The so-called optimal *value function* V^* finally yields the agent's subjective value of being in a certain belief state. Many POMDP solving approaches compute or approximate this function using dynamic programming updates of sub-optimal value functions and derive from the optimal (or good enough) value function the optimal (or good enough) action policy (e.g., [4, 13]). There are also algorithms which search the space of policies directly for the optimal policy (e.g., [8]). The latter type of algorithms nevertheless also requires to know the corresponding value functions of policies, in order to evaluate policies and to single out the optimal policy (or a good enough approximation). The following recursive definition of V^* is called the dynamic programming equation of the POMDP (γ is a discount factor):

$$V^*(b) = max_{\alpha \in A_0}(r(b, a) + \gamma \Sigma_{o \in O} Pr(o|b, a) V^*(\tau(b, a, o))) \tag{17}$$

Unfortunately, the belief-MDP is over a continuous state space, which poses various problems. But fortunately, POMDP solvers can exploit the fact that the

MDP for which the optimal value function is a solution is a converted POMDP, a fact which yields certain useful properties of the function.

4 Modal-Logical Constraining of POMDPs

We assume a given POMDP and a knowledge base (KB) of PD_eL assertions. The task of combining these two in order to retrieve a new, normatively and assertively constrained POMDP is twofold:

- Obtaining a constrained belief estimator from prior knowledge in the KB and
- pruning the set of potentially optimal action policies when solving the POMDP in order to observe the the norms encoded in the KB.

4.1 Assigning Propositions to Markovian States

The states of a POMDP don't tell us anything about the values of the propositional variables in the respective states. In contrast, the KB basically tells us which propositions hold after a certain program has terminated. A Kripke model $\mathfrak{K}$ is defined by $\mathfrak{K} = (K, m_{\mathfrak{K}}, \models)$, with $(K, m_{\mathfrak{K}})$ being the Kripke frame consisting of the set of world states K, and the meaning of each atomic formula and each atomic action, given as a mapping $m_{\mathfrak{K}}$ of this formula/action to a subset of the world states (that is, the states where the formula holds) or set of pairs of world states, respectively (that is, the "input state" which is mapped to the "output state" via an action). $m_{\mathfrak{K}}$ can be extended inductively to work with any formula and complex actions (programs) too.

Formally:

$$m_{\mathfrak{K}}(\psi) \subset K \; \textit{foreach} \; \psi \in \Phi, \; \textit{and} \tag{18}$$

$$m_{\mathfrak{K}}(\alpha) \subset K \times K \; \textit{foreach} \; \alpha \in \textit{Act} \tag{19}$$

With this, we can define the semantics of PD_eL formulas based on the semantics of PDL [2], like:

$$\sigma \models [\alpha]\psi \Leftrightarrow_{def} \forall \sigma' : \textit{if} \; (\sigma, \sigma') \in m_{\mathfrak{K}}(\alpha) \; \textit{then} \; \sigma' \models \psi \tag{20}$$

Theoretically, we could use this semantics directly i) to update the POMDP state transition matrix with definite transitions, ii) to set an element (subjective state probability) of a POMDP belief state to zero if the respective state would be logically impossible, and iii) to gain knowledge about the values of proposition variables after each belief update. In case i) and ii), the given POMDP is treated as possibly partially invalid, and the invalid parts are precisely those which are "overwritten" with definite prior knowledge deductively obtained from the KB. We treat approaches iii) and i) as mutually exclusive: iii) "believes" the result of the POMDP state estimation, whereas i) possibly extinguishes a result of the

probabilistic state estimation.

ii) is expressed as follows:

$$\textit{if } \sigma \models [a]\psi, a \in A_0 \textit{ and } \sigma' \not\models \psi \textit{ then } b'(\sigma') = 0, \tag{21}$$

with $b' = \tau(b, a, o)$, for any observation o. In addition, a re-normalization of the probabilities of the other states is required, so that the sum of the probabilities becomes 1 again - which means that it is not possible to make all states impossible states at the same time!

For iii), we need to extend our notion of belief states to *logically-annotated belief states* $b_\Phi : S \times ([0;1] \times \Phi)$. Then we have the rule

$$\textit{if } \sigma \models [a]\psi, a \in A_0 \textit{ then } b'_\Phi(\sigma') = (\frac{O(\sigma', a, o) \sum_{\sigma \in S} T(\sigma, a, \sigma') b(\sigma)}{Pr(o|a, b)}, \psi), \tag{22}$$

for any observation o.

The retrieval of formal knowledge about a possible state during state estimation can be useful for the acting agent, provided it can interpret the logical annotations.

i) can be expressed using the rule

$$\textit{if } \sigma \models [a]\psi, a \in A_0 \textit{ and } \sigma' \not\models \psi \textit{ then } T(\sigma, \alpha, \sigma') = 0 \tag{23}$$

(again, this would require normalization of the belief state to make it represent a probability distribution).

Practically, it would make sense to consider only formulas which hold in *all* states:

$$\models \psi \Leftrightarrow_{def} \forall \sigma \in K : \sigma \models \psi$$

The constraining of belief updates does then not depend on the respective previous states anymore.

But still the approaches i)-iii) have obviously two shortcomings: firstly, we do not know the mapping of Markovian states to states in K. Secondly, they work only with atomic actions.

The first problem could be solved by considering POMDPs with logically-annotated Markovian states (not to be confused with the logically-annotated belief states above). If we would annotate a subset of the states with a set of formal assertions each, we could nullify those parts of the current belief state which are logically inconsistent w.r.t. the KB. Let $\phi : S \rightarrow 2^\Phi$ be a function

which maps a Markovian state to a (possibly empty) set of assertions which are *known* by the agent to hold in this state. Then

$$\textit{if } \models [a]\psi, a \in A_0 \textit{ and } \models \neg(\psi \wedge \bigwedge_{f \in \phi(\sigma')}) \textit{ then } b'(\sigma') = 0, \tag{24}$$

with $b' = \tau(b, a, o)$, for any observation o and any previous belief state b.

Getting rid of the second shortcoming would be a bit more tricky: we deal with state transitions instead of action histories: each Markovian state is a sufficient statistics in the sense that the probability distribution of successor states depends only on this state (and the current action and observation) and not on any preceding states or action history.

4.2 Constraining the Optimal Action Policy using Norms

Each atomic actions sequence which leads to a world state where the special proposition V holds should be removed from the set of candidates for the optimal action policy, or the value (utility) of such states should be reduced. With this, it becomes more unlikely than otherwise (but not necessarily impossible) that the solution of the POMDP tells the agent to run into a norm-violating state.

In the most simple case, the agent is forbidden to take a single atomic action. In POMDP terms, this can be taken into account by reducing the respective reward of performing this action in any state:

if $\models Fa, a \in A_0$ *then* $\forall \sigma \in S : R(\sigma, a) = \varrho$. Here, the reward is simply set to some negative value ϱ in order to make action a less desirable.

We could alternatively annotate all states but one which are reachable via a with $\neg V$. However, a POMDP does not allow us to make a certain action always lead to an "impossible" Markovian state.

The general case of prohibited (obligatory, allowed) complex actions is significantly more complicated:

The sequences of atomic actions described by a certain action expression can be represented as so-called *s-traces* (*synchronicity traces*) [1]. To represent the set of all s-traces for a certain action expression, we use the notation $[[\alpha]]$. The exact definition of this function can be found in [1]. Each s-trace $s \in [[\alpha]]$ is a sequence $S_1, ..., S_n, ...$ of so-called *synchronicity sets* (*s-sets*) S_i. A single s-set if a subset of Act_0. Intuitively, a single s-set represents a number of atomic actions which are performed concurrently (if the set contains more than one action), or a single atomic action. We call each possible sequence of atomic actions within $[[\alpha]]$ a *run* of α.

To enact the prohibition of a complex action α using its s-traces, we propose the following alternative approaches:

1. Modify the optimal policy directly, in order to make the execution of any action sequence within $[[\alpha]]$ impossible.
2. Modify (decrease) the values of the belief states which are reachable using action sequences in $[[\alpha]]$.
3. Modify (lower) those vectors which contribute to the value function computed during value or policy iteration and which represent an action within an action sequence in $[[\alpha]]$ (see below).

Both policy search and value function search algorithms for solving a POMDP require the computation of value functions (cf. Section 3), from which the optimal policy can be derived directly. For approach 1, we assume that the optimal policy (ignoring norms) is already given, in form of a finite-state machine (FSM). A FSM can always be used to represent the optimal behavioral policy of a *finite-horizon* POMDP, which appears to be a reasonable restriction in our context [8].

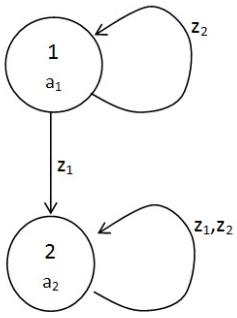

Fig. 1. A simple FSM representing a policy

Figure 1 depicts a FSM which represents a policy for some POMDP as follows: each node (FSM state, labeled with a number inside of the respective node) is annotated with an action a_i which the agents takes in this FSM state. If the policy is optimal, this action is optimal in the respective FSM state. State 1 is the start state. Each arc represents a possible observation z_i following the respective action and leads to a new FSM state. There can be more FSM states than world states.

Assume we have $\models F(a_1; a_1); a_2$. The s-trace of the forbidden program would then simply consist of a single, deterministic run of atomic actions. Removing this sequence from the FSM could yield the new FSM depicted in Figure 2 (with the dashed arc not being part of the FSM). Of course, this FSM is just one

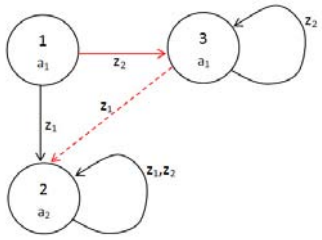

Fig. 2. An updated FSM representing a policy with forbidden paths

among many possible updated FSMs. The major shortcoming of approach 1 is obviously that the agent is not prevented from starting a forbidden action sequence. The sequence is simply cut off before it finishes and makes the agent stop then. Re-directing the agent to some random state instead appears not to be an improvement.

Approach 2 requires us to keep track of actions during the iterative belief state updates, provided the used POMDP solving algorithm allows us to do this. We assume again that $\models F(a_1; a_1); a_2$. The set of belief states the agent might end in after performing run $a_1 \circ a_1$ is computed as $B_{a_1 \circ a_1} = \{\tau(\tau(b_s, a_1, o_1), a_1, o_2)$ for all possible observations o_i and initial belief states $b_s\}$.

After(!) having computed the optimal value function, the agent can incrementally update its belief state and compute at each step the optimal action from the optimal value function. Should the agent run into one of the belief states in $B_{a_1 \circ a_1}$, and the optimal action policy (ignoring norms) suggests to take action a_2 next, it could, as with approach 1, avoid doing so. Again, this approach is rather unconvincing. To be more interesting appears a reduction of the values of the belief states $\{\tau(\tau(\tau(b_s, a_1, o_1), a_1, o_2), a_2, o_3)\}$ during the search for the optimal value function. This way, the agent is more likely to be prevented of running into a forbidden sequence of acting, since the value of a certain state includes the values of succeeding states.

Finally, approach 3 makes use of the fact that each state of a (possibly yet sub-optimal) FSM representing a policy during the search for an optimal policy (using one of the POMDP solvers which search the policy space directly for the optimal policy) corresponds to a vector $v^i(s)$ of the piecewise linear and convex value function which can be computed from this FSM under certain conditions [8]. The value function is the solution of the following system of equations, with one equation for each pair of FSM state i and Markovian state σ:

$$v^i(\sigma) = R(\sigma, a(i)) + \gamma \sum_{\sigma', z} Pr(\sigma' | \sigma, a(i)) Pr(z | \sigma', a(i)) v^{l(i,z)(\sigma')} \qquad (25)$$

Performing a forbidden sequence of atomic actions as determined by the FSM and observations yields a resulting FSM state which corresponds to exactly one of these vectors v^i, which is associated with the optimal action in this state. Lowering this vector, i.e., modifying the value function at this place, would lower the value of this FSM, and would, as we assume, lead in the next policy search step to a FSM which is closer to norm-observing behavior. The advantage of this approach is that from a modified set of vectors (which is then in addition also improved by a dynamic programming update) a new FSM can be constructed very easily [8]. However, this approach would require extensive experimental evaluation in order to judge whether it would actually make sense in a concrete scenario.

5 Conclusion

In this extended abstract we have proposed various initial means for the embedding of knowledge about norms and formal knowledge in general into POMDPs, hoping to initiate a new line of future research. Although we have hopefully given some initial insight into the challenge, a lot of work remains to be done:

- Identification of application scenarios which are on the one hand rich enough to allow for a more or less realistic normative system, but which are on the other hand still approachable by contemporary POMDP solver.
- Detailed empirical and theoretical analysis of the constraining task, with a detailed comparison of the proposed and further ways of incorporating norms into POMDPs and stochastic decision processes in general.
- Detailed empirical and theoretical analysis of how the constraining affects the POMDP solving algorithm.
- Consideration of more complex kinds of norms, such as norm hierarchies and preferences among norms.

References

1. J.-J. Meyer. A Different Approach to Deontic Logic: Deontic Logic Viewed as a Variant of Dynamic Logic. Notre Dame Journal of Formal Logic, 29, 1988.
2. D. Harel, D. Kozen and J. Tiuryn. Dynamic Logic. MIT Press, 2000.
3. P. Blackburn, J. van Benthem, F. Wolter (Eds.). Handbook of Modal Logic. Elsevier, 2006.
4. L. P. Kaelbling, M. L. Littman, A. R. Cassandra. Planning and Acting in Partially Observable Stochastic Domains. Artificial Intelligence 101(1-2): 99-134, 1998
5. K. Wagstaff, C. Cardie. Clustering with Instance-level Constraints. In Proceedings of the 17th International Conference on Machine Learning (ICML 2000), 2000.
6. C. Boutilier, R. Reiter, M. Soutchanski, S. Thrun. Decision-Theoretic, High-level Agent Programming in the Situation Calculus. AAAI 2000, 2000.
7. A. Farinelli, A. Finzi, Th. Lukasiewicz. Team Programming in Golog under Partial Observability. In Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI 2007). AAAI Press / IJCAI, 2007.

8. E. Hansen. Solving POMDPs by Searching in Policy Space. In Proceedings of the Fourteenth International Conference on Uncertainty In Artificial Intelligence (UAI-98), 1998.
9. F. Fischer, M. Nickles. Computational Opinions. In Proceedings of the 17th European Conference on Artificial Intelligence (ECAI-06), IOS Press, 2006.
10. G. Boella, M. Singh, G. Pigozzi, H. Verhagen (Eds.). Proceedings of the Third International Workshop on Normative Multiagent Systems (NorMAS), 2008.
11. I. Davidson, S. S. Ravi. The complexity of non-hierarchical clustering with instance and cluster level constraints. Data Mining and Knowledge Discovery 14(1), 2007.
12. L. Getoor, B. Taskar (Eds.). Introduction to Statistical Relational Learning. The MIT Press, 2007.
13. M. Littman. Solving Partially Observable Markov Decision Processes via VFA. In J. A. Boyan, A. W. Moore, R. S. Sutton (Eds.), Proceedings of the Workshop on Value Function Approximation, Machine Learning Conference 1995, Technical report CMU-CS-95-206, 1995.
14. M. Fischer, R. Ladner. Propositional Dynamic Logic of Regular Programs, Journal of Computer and System Sciences, 18: 194-211.