Propositional Logic and One-Stage Decision Making

Hélène Fargier **Jérôme Lang**
IRIT
Université Paul Sabatier
31062 Toulouse Cedex (France)
{fargier, lang}@irit.fr

Pierre Marquis
CRIL
Université d'Artois
62307 Lens Cedex (France)
marquis@cril.univ-artois.fr

Abstract

In this paper, a pure propositional framework for representing qualitative, one-stage decision problems under uncertainty is presented. The representation is based on a partition of the propositional variables between state variables – observable or not – and decision variables, and a distinction between pieces of knowledge, constraints, and goals. Effects of decisions are computed using dependency-based update and the Modified Possible Models Approach (MPMA). Various assumptions about the structure of the problem (especially observability), and the way of computing the consequence of a decision, are considered. We are specifically interested in finding out decision policies, i.e., determining for each possible observation a decision that ensures to reach a goal state if taken (whatever the unobservable variables). The existence of a solution policy is investigated from the point of view of computational complexity. The search for a policy is shown to be related to the resolution of the validity problem for some quantified boolean formulas (QBF), and to the existence of some prime implicants modulo a formula.

1 Introduction

Informally, decision making under uncertainty consists in finding suitable actions (possibly in sequence, possibly conditionally on observations) so as to be likely enough to reach a satisfactory state.

In classical decision theory (e.g., [33]) the data of a given problem are generally represented in an extensive, space-consuming form by listing all states, all actions, all possible consequences, etc. This is where AI (and more specifically logic) has a role to play, because it has contributed to the development of representation languages which can encode a decision problem in a much more structured and compact way. Actually, logic can be used in (at least) three different ways. The first one is the description of the system (its initial state, the available decisions and their effects) using expressive knowledge representation languages (especially for reasoning about actions). The second one is the description of the preferences of the agent, through preference representation languages. The third one is the actual computation of decision policies using automated reasoning techniques (such as theorem proving, model finding, abduction).

In this paper, we make the following assumptions about the decision process and its representation:

1. *structured decisions.* Not only the *state* space is represented in a compact, structured way, but so is the *decision* (or *action*) space. Unlike in Markov Decision Processes and most planning formalisms, decisions consist of simultaneous truth assignments of a number of decision variables (consider for instance the cases where assigning a decision variable to *true* or *false* consists in pushing a button or not, or, taking or not a medicine). To express that some particular assignments are (physically) impossible, the data include a set of *constraints* bearing on decision variables.
2. *partial observability* of the initial state of the system. Specific cases of it are full observability, and, at the opposite side, unobservability.
3. *nondeterministic effects of decisions.* Effects of decisions are encoded using the recent framework for belief update named *Modified Possible Models Approach* (MPMA) [10], which offers a much more satisficing handling of disjunctive updates than Winslett's standard PMA [35]. An additional strong point of the MPMA compared with

the PMA is its complexity (which is an extra-result offered by this paper): coNP-complete versus Π_2^p-complete [12].

4. *purely qualitative expression of knowledge and goals.* Everything is encoded in pure propositional logic. Especially, binary preferences are considered, only: states are partitioned among goal states and non-goal states and the purpose of the agent is to find out decisions enabling to reach a goal state when taken. No graded uncertainty about the possible outcomes of an action (e.g., a probability distribution over the resulting states) is taken into account.

5. *one decision stage.* This simplification not only makes the exposition simpler, but it is also practically relevant because (i) many problems involve only one decision stage and (ii) our framework can be easily extended (see in Conclusion) to multi-stage decision *without observability.*

Points (1) (2) and (3) above offer enough flexibility to make our framework practically relevant. Especially, propositional STRIPS and several generalizations of it can be viewed as specific instances of this framework. Points (4) and (5) can be considered quite restrictive; nevertheless, (4) enables a pure propositional logic framework for decision making; hence it is possible to take advantage of various well-known computational techniques of this framework, like prime implicants algorithms. Another motivation for (4) and (5) is the necessity to focus on some specific cases of general decision making so as to identify precisely its sources of complexity (the existence of a plan in the propositional STRIPS framework is already PSPACE-complete [5]).

The problem of qualitative decision making considered in this paper is the following: *is there a policy assigning a decision to each possible observation that ensures that a goal state will be reached?* The purpose of this paper is to study both the existence of such a policy (especially from a computational complexity point of view) and the practical computation of policies fulfilling the above condition *as much as possible.* The search for a solution policy for a qualitative decision problem is shown related to the validity problem for *quantified boolean formulas*, or QBF (studied mainly in the area of complexity theory [34] [29]). It is also related to the existence of some specific prime implicants modulo a formula. Based on this connection, an algorithm for computing policies is presented.

2 Background and notations

2.1 Propositional logic

Let PS be a finite set of propositional variables. $PROP_{PS}$ is the propositional language built up from PS and the connectives in the usual way. For every $X \subseteq PS$, $PROP_X$ denotes the sublanguage of $PROP_{PS}$ generated from the variables of X, only. $\top$ denotes the boolean constant that is always true, and $\bot$ the boolean constant that is always false. For every formula Σ of $PROP_{PS}$, $Var(\Sigma)$ is the set of variables occurring in Σ. $\Sigma[x \leftarrow \top]$ (resp. $\Sigma[x \leftarrow \bot]$) denotes the formula from $PROP_{PS}$ obtained by substituting in a uniform way the variable $x \in PS$ by the boolean constant $\top$ (resp. $\bot$) in Σ. Full instantiations of variables of $X \subseteq PS$ (resp. PS) (worlds) are denoted by $\vec{x}$ (resp. $\vec{\omega}$). Sometimes we identify the complete assignment $\vec{x}$ with the usual formula of $PROP_X$ formed by taking the conjunction of all X-literals as assigned by $\vec{x}$. 2^X denotes the set of all possible assignments of variables of X. If X and Y are two disjoint subsets of $PROP_{PS}$ then $(\vec{x}, \vec{y})$ is the assignment of $2^{X \cup Y}$ that assigns any variable of X like $\vec{x}$ does and any variable of Y as $\vec{y}$ does. We shall also identify finite sets of formulas with the conjunctions of their elements each time this will be convenient.

The set $PI_\Sigma^X(\Phi)$ of prime implicants modulo a formula Σ of a formula Φ over $PROP_X$ is defined (up to logical equivalence) by $PI_\Sigma^X(\Phi) = PI^X(\Sigma \Rightarrow \Phi) \setminus PI^X(\neg\Sigma)$, where $PI^X(\Phi) = PI(\Phi) \cap PROP_X$ and $PI(\Phi)$ ($= PI_\top^{PS}(\Phi)$) denotes the set of prime implicants of Φ. For instance, if $\Sigma = \{a \vee b, (a \wedge c) \Rightarrow e, d \Rightarrow e\}$, then $PI_\top^{\{a,b,c,d,e\}}(\Sigma) = \{a \wedge e, a \wedge \neg c \wedge \neg d, b \wedge e, b \wedge \neg c \wedge \neg d, \neg a \wedge b \wedge \neg d\}$; $PI_\top^{\{a,b,c,d,e\}}(\neg\Sigma) = \{\neg a \wedge \neg b, a \wedge c \wedge \neg e, d \wedge \neg e, \neg b \wedge c \wedge \neg e\}$; $PI_\Sigma^{\{a,b,c,d\}}(e) = \{a \wedge c, d, \neg b \wedge c\}$; $PI_\Sigma^{\{a,c\}}(e) = \{a \wedge c\}$; $PI_\Sigma^{\{a,b\}}(e) = \emptyset$; $PI_\Sigma^{\{a,b,c,d\}}(a \vee b) = \{\top\}$.

2.2 Computational complexity and QBF

In this paper we refer to some complexity classes above NP and coNP, details about which can be found in [29]. We assume that the classes P, NP and coNP are known to the reader. $\Sigma_2^p = \mathsf{NP}^{\mathsf{NP}}$ is the class of all languages recognizable in polynomial time by a nondeterministic Turing machine equipped with an NP oracle telling in unit time whether a given propositional formula is satisfiable or not; $\Pi_2^p = \mathsf{co}\Sigma_2^p$; $\Pi_3^p = \mathsf{coNP}^{\Sigma_2^p}$. The canonical Σ_2^p-complete problem is 2-QBF (where QBF stands for "quantified boolean formula"). An instance of 2-QBF consists of a triple $\langle A, B, \Phi \rangle$ where A and B form a partition of the set of variables appearing in Φ. Such a triple is a *positive* instance of 2-QBF iff $\exists \vec{a} \in 2^A$ such that $\forall \vec{b} \in 2^B$, $(\vec{a}, \vec{b}) \models \Phi$. A characterization of the positive instances of the canonical Π_2^p-complete problem 2-$\overline{\text{QBF}}$ is obtained by reversing the quantifiers, namely: $\forall \vec{a} \in 2^A \; \exists \vec{b} \in 2^B$ such that $(\vec{a}, \vec{b}) \models \Phi$. Finally, an instance of the canonical Π_3^p-complete problem 3-$\overline{\text{QBF}}$ is a 4-uple $\langle A, B, C, \Phi \rangle$, where $\{A, B, C\}$ is a partition of $Var(\Phi)$, and the positive ones are such that $\forall \vec{a} \in 2^A \; \exists \vec{b} \in 2^B \; \forall \vec{c} \in 2^C$, $(\vec{a}, \vec{b}, \vec{c}) \models \Phi$. The following easy result shows that positive instances of 2-QBF and of 2-$\overline{\text{QBF}}$ can be characterized using prime implicants modulo a formula:

Proposition 1 (2-QBF and prime implicants)

- $\langle A, B, \Phi \rangle$ *is a positive instance of 2-*QBF *if and only if* $PI^A(\Phi) \neq \emptyset$.
- $\langle A, B, \Phi \rangle$ *is a positive instance of 2-*$\overline{\text{QBF}}$ *if and only if* $PI^A(\neg\Phi) = \emptyset$.

Accordingly, with $\Sigma = \{a \vee b, (a \wedge c) \Rightarrow e, d \Rightarrow e\}$, $\langle \{b, c, e\}, \{a, d\}, \Sigma \rangle$ is a positive instance of 2-QBF (because $PI_{\top}^{\{b,c,e\}}(\Sigma) = \{b \wedge e\}$) while $\langle \{a, b, c\}, \{d, e\}, \Sigma \rangle$ is not (because $PI_{\top}^{\{a,b,c\}}(\Sigma) = \emptyset$). $\langle \{a\}, \{b, c, d, e\}, \Sigma \rangle$ is a positive instance of 2-$\overline{\text{QBF}}$ while $\langle \{a, b\}, \{c, d, e\}, \Sigma \rangle$ is not. Moreover, $\langle \{a\}, \{b, c, e\}, \{d\}, \Sigma \rangle$ is a positive instance of 3-$\overline{\text{QBF}}$ while $\langle \{a\}, \{b, c\}, \{d, e\}, \Sigma \rangle$ is not.

2.3 Dependency-based update and the MPMA

We conclude this section with some basics about dependency-based update [16] and the Modified Possible Models Approach (MPMA) [10]. Here, the syntactic presentations of updated knowledge bases do not matter, hence we define them model-theoretically by sets of models. Let $P \subseteq PS$, $\vec{\omega}$ a world and α a formula. The *MPMA-update* [10] of $\vec{\omega}$ by α w.r.t. P, denoted by $\vec{\omega} \star^P \alpha$, is the set of worlds $\vec{\omega}'$ s.t. $\vec{\omega}' \models \alpha$ and for every propositional variable v not in P, $\vec{\omega}$ and $\vec{\omega}'$ assign the same truth value to v. If KB is a formula from $PROP_{PS}$, then the MPMA-update $KB \star^P \alpha$ of KB by α w.r.t. P is defined by $KB \star^P \alpha = \bigcup_{\vec{\omega} \models KB} (\vec{\omega} \star^P \alpha)$. The set of variables P is allowed to vary. At least four specific choices are worth considering for P:

1. one extreme case is $P = PS$, which leads to forgetting everything about KB. Namely, $KB \star^{PS} \alpha \equiv \alpha$;
2. the other extreme case is $P = \emptyset$, which comes down to classical conjunction: $KB \star^{\emptyset} \alpha \equiv KB \wedge \alpha$;
3. as proposed by Herzig [16], P may be the set of variables $Var(\alpha)$ mentioned in α – this appears reasonable because variables not appearing in α can be considered having nothing to do with α – thus it is not necessary to forget about them before adding α;
4. however, it might be the case that some variables appear (syntactically) in α while being (semantically) irrelevant to α. Doherty et al. [10] propose the choice of $P = DepVar(\alpha)$, namely the set of variables α is dependent on, defined by: $p \in DepVar(\alpha)$ iff p occurs in any formula α' logically equivalent to α, or, equivalently, $p \in DepVar(\alpha)$ iff $\alpha[p \leftarrow \top]$ is not equivalent to $\alpha[p \leftarrow \bot]$ [10] [22] [21] [1].

Doherty et al. [10] give the following syntactical characterization of the MPMA: if $P = \{p_1, \dots p_n\}$, then $\exists p_i \alpha = \alpha[p_i \leftarrow \top] \vee \alpha[p_i \leftarrow \bot]$ and $\exists P.KB$ stands for $\exists p_1 \dots \exists p_n KB$. The following holds: $KB \star^P \alpha \equiv \alpha \wedge \exists P.KB$. Thus, $KB \star^P \alpha$ is equivalent to a (simple) quantified boolean formula (with some free variables!). Such a formula can be turned into a classical one using variable elimination algorithms (e.g., directional resolution [9]), but there is no guarantee that the size of the resulting formula is polynomial in the input size (and the existence of such a rewriting is unlikely since it would make the polynomial hierarchy to collapse at the first level [21]). Interestingly, in the case where $P \supseteq DepVar(\alpha)$ (e.g., in cases 1., 3. or 4. above), the above characterization of MPMA-update through quantified boolean formula shows that $KB \star^P \alpha$ is consistent as soon as KB is consistent and α is consistent.

3 Qualitative one-stage decision problems

3.1 Knowledge, goals, observations, and decisions

A qualitative decision problem intuitively consists of a partition of the variables of interest between decision and state variables (where each state variable is either observable or unobservable), a knowledge base K describing the initial state of the world, a set of constraints C restricting the different decisions available to the agent, the description of the effects of decisions EFF, and a set of goals G.

[1] Contrariwise to Doherty et al. [10], we do not think that such a characterization is impractical from a computational point of view. Indeed, computing $DepVar(\alpha)$ can be achieved through "only" a linear number of calls to an NP oracle. Contrastingly, the characterization presented in [10] requires the computation of the set of prime implicants of α, whose cardinality is exponential in the size of α in the worst case.

Definition 1 (qualitative decision problem) *A* qualitative decision problem (QDP) *is a 7-uple* $\mathcal{P} = \langle DecVar, ObsVar, UnobsVar, K, C, G, EFF\rangle$ *where:*

- $\{DecVar, ObsVar, UnobsVar\}$ *is a partition of* PS. *Variables of* $DecVar$ *(resp.* $ObsVar$, $UnobsVar$*) are* decision variables *(resp.* observable state variables, unobservable state variables*).* $StateVar = ObsVar \cup UnobsVar$ *is the set of* state variables*.*
- K *and* G *are finite consistent sets of formulas on* $StateVar$*.* C *is a consistent set of formulas on* $DecVar$*.*
- EFF *is a finite set of triples* $\langle\varphi, \delta, \psi\rangle$ *where* φ *and* ψ *are consistent formulas on* $StateVar$ *and* δ *is a consistent formula on* $DecVar$*.*

Decision variables are under the control of the agent (the latter has the power to fix their truth value). Such variables are sometimes called *controllable* [1] [23]. A complete assignment $\vec{d}$ of all decision variables is called a *decision*. The decision space 2^{DecVar} is denoted by D.

State variables describe the state of the world (before and after the decision is taken). They are not *directly* under the control of the agent but some of them may be influenced by the agent. They are partitioned into two subclasses: *observable (state) variables* are observed, i.e., their truth value is known before the agent acts, whereas the truth value of *unobservable (state) variables* is not directly observable by the agent. A complete (resp. partial) assignment $\vec{o}$ of observable variables is called a complete (resp. partial) *observation*. The set of all (complete or partial) observations (*observation space*) is denoted by O. $\vec{u}$ denotes a complete assignment of unobservable variables; the set of all such assignments is denoted by U. If $\vec{o}$ is a complete observation and $\vec{u}$ is a complete assignment of unobservable variables, then $\vec{s} = (\vec{o}, \vec{u})$ is a complete assignment of all state variables. S denotes the set of all such assignments (called *states*).

K, bearing on state variables only, describes the *initial state* of the world. Of course, in the general case, only partial knowledge about this state is available (i.e., K may easily have several models on $StateVar$).

EFF describes the *effects of decisions*. Each triple $\langle\varphi, \delta, \psi\rangle$ of EFF has the following meaning: "if the initial state (before the decision is taken) satisfies φ and the decision undertaken satisfies δ, then the resulting state satisfies ψ". Clearly, these triples used for representing effects of actions are similar to the transitions used by Cordier and Siegel in [7].

Formulas of C are (physical) *constraints* restricting the allowed combination of decision variables: if such a combination $\vec{d}$ does not satisfy C, then the corresponding decision is not available to the agent[2].

Formulas of G are *goals*. They are specified by the agent and express his/her preferences about the consequences of his/her decisions. They involve state variables only – without loss of generality since decision variables upon which the agent has preferences *can be duplicated* by as many state variables, through the expansion of EFF with additional effects rules of the form $\langle\top, d, d_{obs}\rangle$ and $\langle\top, \neg d, \neg d_{obs}\rangle$ for every $d \in DecVar$.

Three particular cases are of interest regarding the variables:

- $ObsVar = \emptyset$: all state variables are unobservable and $\mathcal{P}$ is a QDP under *unobservability* (UQDP).
- $UnobsVar = \emptyset$: all state variables are observable and $\mathcal{P}$ is a QDP under *full observability* (FQDP).
- $UnobsVar = \emptyset$ and moreover there is only one possible state consistent with K: the initial state is completely known in advance (*off-line*) – to be distinguished from the more general case of full observability where the initial state is fully observed *on-line*. $\mathcal{P}$ is a QDP under *complete initial knowledge*.

We consider particular cases regarding constraints:

- *Full executability* corresponds to $C = \emptyset$ and means that any combination of decision variables is allowed, which makes $2^{|DecVar|}$ available decisions.
- *Atomicity* corresponds to the case where C contains $\neg(d \wedge d')$ for all $d, d' \neq d \in DecVar$. Decisions correspond then to atomic actions (which is the classical assumption in planning and in Markov Decision Processes). Under atomicity, only $|DecVar| + 1$ decisions are available: every decision that satisfies more than one $d_i \in DecVar$ is not consistent with C.

We also consider particular cases regarding the effects of decisions:

[2] More generally, we could have allowed *conditional* constraints, i.e., pairs $\langle\varphi, \gamma\rangle$ (where φ involves state variables) meaning "if the initial state satisfies φ, then the decision taken must satisfy γ". For the sake of simplicity, we omit this possibility.

- STRIPS: (i) for each $\langle\varphi,\delta,\psi\rangle$ in EFF, φ and ψ are terms on $StateVar$; (ii) K is a complete term on $StateVar$ (i.e., each variable of $StateVar$ occurs once in K, either positively or negatively); (iii) G is a term on $StateVar$; (iv) EFF satisfies atomicity. Clearly enough, due to (ii), only full observability makes sense under the STRIPS restriction.

- STRIPS$^+$: (i) as above and nothing else (K and G are any formulas on $StateVar$ and atomicity is not required).

- *qualitative influence diagrams* (QID): for each $\langle\varphi,\delta,\psi\rangle$ in EFF, ψ is a literal [3].

For the sake of simplicity, ramifications are handled explicitely in this paper. They could be handled automatically using either explicit dependencies as in [16] or integrity constraints as in [10]. We omit this in order to simplify the exposition.

Example 1 (aunt Agatha's living-room, revisited).

$$
\begin{array}{lll}
ObsVar & = & \{\text{cold, summer, stuffy}\};\\
UnobsVar & = & \emptyset;\\
DecVar & = & \{\text{heating, (window-)open}\};\\
K & = & \{\text{summer} \Rightarrow \neg\text{cold}\};\\
C & = & \emptyset;\\
EFF & = & \{\ \langle\text{cold, heating} \wedge\neg\text{open}, \neg\text{cold}\rangle,\\
 & & \quad\langle\top, \text{open}, \neg\text{stuffy}\rangle,\\
 & & \quad\langle\neg\text{summer, open, cold}\rangle\};\\
G & = & \{\neg\text{cold}, \neg\text{stuffy}\}.
\end{array}
$$

K and G do not need any explanation; let us briefly explain the effects items:

- $\langle$cold, heating $\wedge\neg$open, $\neg$cold$\rangle$ means that if the room is initially cold and if the decision taken includes (among other things) putting the heating on and closing the window, then in the resulting state the room will not remain cold.

- $\langle\top$, open, $\neg$stuffy$\rangle$ means that opening the window makes the room non-stuffy. We do not assume here that closing the window of a non-stuffy room does make it stuffy. If we want to represent the fact that closing the window *may* make the room stuffy (and may not), then one way to do it is adding the effect rule $\langle\top$, $\neg$open, stuffy $\vee$lucky$\rangle$, where "lucky" is a new state variable, or alternatively, the effect rule $\langle$lucky, $\neg$open, $\neg$stuffy$\rangle$, where "lucky" is a new *unobservable* state variable. Another way would consist in adding an explicit dependency (as in [16]) telling that closing the window influences the variable "stuffy".

- $\langle\neg$summer, open, cold$\rangle$ means that if it is not summertime, opening the window will make the room cold.

3.2 States, decisions and consequences

Let us now define the following notions of possible observation, possible state, available decision and consequence of a decision on a state given a QDP $\mathcal{P}$.

Definition 2 (possible observations)
An observation $\vec{o}$ is possible *(w.r.t. $\mathcal{P}$) iff $K\wedge\vec{o}$ is consistent. The set of all possible observations is denoted by $PossObs(\mathcal{P})$.*

Definition 3 (possible states)
A state $\vec{s}$ is possible *(w.r.t. $\mathcal{P}$) iff $\vec{s}\models K$. The set of all possible states is denoted by $PossStates(\mathcal{P})$.*

Definition 4 (available decisions) *A decision $\vec{d}$ is* available *(w.r.t. $\mathcal{P}$) iff $\vec{d}\models C$. The set of all available decisions is denoted by $AvDec(\mathcal{P})$.*

When no ambiguity is possible about the underlying QDP $\mathcal{P}$, the simplified notations $PossObs$, $PossStates$, $AvDec$ will be used instead of, respectively, $PossObs(\mathcal{P})$, $PossStates(\mathcal{P})$, $AvDec(\mathcal{P})$.

Definition 5 (consequences of $\vec{d}$ on $\vec{s}$)
Given a QDP $\mathcal{P}$, let $\vec{d}\in AvDec$ and $\vec{s}\in PossStates$.

- $Collect(\vec{d},\vec{s}) = \bigwedge\left\{\psi \;\middle|\; \begin{array}{l}\langle\varphi,\delta,\psi\rangle\in EFF \textit{ such}\\ \textit{that } \vec{s}\models\varphi \textit{ and } \vec{d}\models\delta\end{array}\right\}$;

- $Cons(\vec{d},\vec{s}) \;=\; \vec{s}\star^{P(\vec{d},\vec{s})} Collect(\vec{d},\vec{s})$ [4] *(here, $Cons(\vec{d},\vec{s})$ is identified with the set of its models).*

Thus, the consequence of a decision $\vec{d}$ on a state $\vec{s}$ is computed in two steps: first, $Collect(\vec{d},\vec{s})$ filters all "active" effect rules and gathers their consequence parts; an effect rule $\langle\varphi,\delta,\psi\rangle$ is active if its precondition φ is satisfied by $\vec{s}$ and its decision part δ is satisfied by

[3] This terminology comes, of course, from influence diagrams [18] where a probability for each state variable, conditioned by the values of the decision and state variables it depends on, is specified. Imposing that only literals appear in the consequence part of decision effects correspond thus to the description of such a conditional probability in a degenerate influence diagram where variables are binary and probabilities are either 0 or 1.

[4] To be more rigorous we should use the notation $Cons_{P(\vec{d},\vec{s})}(\vec{d},\vec{s})$ instead of $Cons(\vec{d},\vec{s})$. In order to avoid heavy notations, we omit the subscript $P(\vec{d},\vec{s})$.

$\vec{d}$. Then, $\vec{s}$ is updated with the formula resulting from the preceding step, using the MPMA. We note that the parameter $P(\vec{d}, \vec{s})$ has been left unspecified. The following four choices are worth considering (the first three choices are also commented in the introductory section):

1. $P(\vec{d}, \vec{s}) = PS$, whatever $\vec{s}$ and $\vec{d}$.
2. $P(\vec{d}, \vec{s}) = Var(Collect(\vec{d}, \vec{s}))$.
3. $P(\vec{d}, \vec{s}) = DepVar(Collect(\vec{d}, \vec{s}))$. As shown in [10], this choice, namely the set of variables on which $Collect(\vec{d}, \vec{s})$ is dependent, works well in practice. Unfortunately, computing $DepVar(Collect(\vec{d}, \vec{s}))$ is NP-hard [22] and this task can not be preprocessed because there are exponentially many formulas $Collect(\vec{d}, \vec{s})$ in the worst case.
4. $P(\vec{d}, \vec{s}) = \bigcup\{DepVar(\psi) | \langle\varphi, \delta, \psi\rangle \in EFF, \vec{s} \models \varphi, \vec{d} \models \delta\}$. This amounts to a preprocessing phase where a dependency set $DepVar(\psi)$ is computed *for every effect rule* $\langle\varphi, \delta, \psi\rangle$ in EFF, and a linear-time postprocessing phase consisting in taking the union of such variables for all fired effect rules. It is not hard to see that in this case $P(\vec{d}, \vec{s})$ is an *upper approximation* of $DepVar(Collect(\vec{d}, \vec{s}))$, i.e., we have $P(\vec{d}, \vec{s}) \supseteq DepVar(Collect(\vec{d}, \vec{s}))$. The equality does not hold in the general case: for instance, if $EFF = \{\langle\top, x, a \vee b\rangle, \langle\top, y, \neg a \vee b\}$, $\vec{s} = a \wedge b$, $\vec{d} = x \wedge y\}$, then $Collect(\vec{d}, \vec{s}) \equiv b$ and $DepVar(Collect(\vec{d}, \vec{s})) = \{b\}$ while $DepVar(a \vee b) \cup DepVar(\neg a \vee b) = \{a, b\}$. However, whether the best choice here for $P(\vec{d}, \vec{s})$ is $\{a, b\}$ or $\{b\}$ is unclear. If $P(\vec{d}, \vec{s}) = \{b\}$, then $\vec{s} \star^{\{b\}} Collect(\vec{d}, \vec{s}) = \vec{s} \star^{\{b\}} b = a \wedge b$, whereas if $P(\vec{d}, \vec{s}) = \{a, b\}$, then $\vec{s} \star^{\{a,b\}} b = b$. Which one is intuitively more satisfactory is left for further research.

Now, in the general case the agent only has a *partial knowledge* about the real world, which leads us to defining the consequence of a decision on an observation.

Definition 6 (consequences of $\vec{d}$ on $\vec{o}$)
Given a QDP $\mathcal{P}$, for any $\vec{d} \in AvDec$ and $\vec{o} \in PossObs$, $Cons(\vec{d}, \vec{o})$ is defined model-theoretically by the set of models $\bigcup\{Cons(\vec{d}, \vec{s}) \mid \vec{s} \models \vec{o}\}$.

Example 1 (continued)
Let us make any of the three assumptions 2, 3, 4 for the choice of $P(\vec{d}, \vec{s})$, and let $\vec{o} =$ cold, $\vec{d} =$ heating $\wedge\neg$open; $\vec{d'} =$ heating $\wedge$open. We have:
- $Cons(\vec{d}, \vec{o}) \equiv$ cold $\wedge\neg$summer.
- $Cons(\vec{d'}, \vec{o}) \equiv$ cold $\wedge\neg$summer $\wedge\neg$stuffy.

For $\vec{d}$, the two consequent states come respectively from the two possible states $\vec{s} =$ (cold, $\neg$summer, stuffy) and $\vec{s'} =$ (cold, $\neg$summer, $\neg$stuffy). Should we have done the choice $P(\vec{d}, \vec{s}) = PS$ then we would have got $Cons(\vec{d}, \vec{o}) \equiv$ cold, because nothing tells *explicitely* that summertime does not come in the meanwhile, even if nothing tells the contrary: implicit persistence is not handled.

3.3 Consistency and determinism

Some global properties can be expected from a QDP in order to ensure that its specification is correct. Especially, it can be expected that each time an observation is possible and a decision is available, the decision can be taken without producing an inconsistency. More formally:

Definition 7 (consistency)
A QDP $\mathcal{P}$ is consistent *iff $\forall \vec{s} \in PossStates$, $\forall \vec{d} \in AvDec$, $Collect(\vec{d}, \vec{s})$ is consistent.*

It is easy to check that the QDP given at Example 1 is consistent.

Consistency guarantees that the description of causal laws (between initial states, decisions and consequent states) is well-founded, i.e., it allows at least one possible consequence for any available decision and any possible initial state[5]. *From now on, we assume that all the QDPs considered are consistent.*

It is important to remark that consistency implies that $Cons(\vec{d}, \vec{s}) \neq \emptyset$ as soon as $P(\vec{d}, \vec{s}) \supseteq DepVar(Collect(\vec{d}, \vec{s}))$, (consequence of Theorem 2 in [10]) – and our four possible choices for $P(\vec{d}, \vec{s})$ are all supersets of $DepVar(Collect(\vec{d}, \vec{s}))$.

QDP CONSISTENCY is the decision problem that consists in determining whether a given QDP is consistent. As the following proposition shows it, checking consistency is computationally a heavy task in the general case.

Proposition 2 (QDP CONSISTENCY)
The complexity of QDP CONSISTENCY *is reported in the following two tables:*

[5] Actually, to a QDP can be associated a causal structure similar to a causal network [8] augmented with decision variables and a goal variable. Our definition of consistency has some similarities (though is not equivalent to) causal independence.

	general case
general case	Π_2^p-complete
QID	coNP-complete
STRIPS$^+$	coNP-complete

	+ atomicity	+ full executability
general case	Π_2^p-complete	Π_2^p-complete
QID	coNP-complete	coNP-complete
STRIPS$^+$	coNP-complete	coNP-complete
STRIPS	in P	-

Another valuable property a QDP can exhibit is determinism:

Definition 8 (determinism)
A QDP $\mathcal{P}$ is deterministic *iff* $\forall \vec{s} \in PossStates$, $\forall \vec{d} \in AvDec$, *$Cons(\vec{d}, \vec{s})$ is a singleton.*

Determinism guarantees that given a possible complete initial state and an available decision, all the state variables are determined once the decision has been made on that state. Clearly enough, determinism entails consistency. Importantly, whether a QDP is deterministic or not *depends on the choice for the set $P(\vec{d}, \vec{s})$ used in the update operation* (contrariwise to consistency).

For instance, in Example 1, the QDP is deterministic with $P(\vec{d}, \vec{s}) = DepVar(Collect(\vec{d}, \vec{s}))$ but nondeterministic if $P(\vec{d}, \vec{s}) = PS$.

Interestingly, when the choice for $P(\vec{d}, \vec{s})$ is 1., 2. or 4., determinism is easier to check than consistency. Let QDP DETERMINISM the problem of deciding whether a given QDP is deterministic or not; we have identified the following complexity result:

Proposition 3 (QDP DETERMINISM)
When $P(\vec{d}, \vec{s})$ is computed in polynomial time, QDP DETERMINISM *is* coNP*-complete.*

Interestingly, under the QID assumption or the STRIPS$^+$ assumption, consistency implies determinism when $P(\vec{d}, \vec{s}) = DepVar(Collect(\vec{d}, \vec{s}))$.

Proposition 4 *Let $\mathcal{P}$ be a QDP whose syntax satisfies the* QID ASSUMPTION *or the* STRIPS$^+$ *assumption. Then, if the choice for $P(\vec{d}, \vec{s})$ is 2., 3. or 4.*[6]*, the consistency of $\mathcal{P}$ implies its determinism.*

Indeed, under the QID assumption when $\mathcal{P}$ is consistent, the truth value of each state variable is determined in $Cons(\vec{d}, \vec{s})$; since this is true for each available $\vec{d}$ and each possible $\vec{s}$, $\mathcal{P}$ is deterministic.

[6]Note that they coincide under the QID assumption when $\mathcal{P}$ is consistent.

4 Solving a QDP

Let us now explain what solving a QDP means.

4.1 Observation covering

Solving a QDP consists in determining, given any possible observation, a decision covering it, i.e., that can ensure the goal to be satisfied when the decision is taken *whatever the values of the unobservable variables.*

This calls for the following notions:

Definition 9 (cover, coverable observation)
Let $\mathcal{P}$ be a consistent QDP. Let $\vec{o} \in PossObs$ and $\vec{d} \in AvDec$. $\vec{d}$ covers *$\vec{o}$ iff $Cons(\vec{d}, \vec{o}) \models G$. When there exists a $\vec{d}$ covering $\vec{o}$, $\vec{o}$ is said* coverable.

A rewriting of the definitions leads to the following characterization for observation covering: $\vec{o}$ is coverable iff $\exists \vec{d} \in D$ s.t. $\forall \vec{s} \models K \wedge \vec{o}$, $\forall \vec{s'} \in Cons(\vec{d}, \vec{s})$, $(\vec{d}, \vec{s'}) \models C \wedge G$. Clearly, this looks like a 2-QBF problem (this is completely clear when we choose $P(\vec{d}, \vec{s}) = PS$). This enables determining the complexity of OBSERVATION COVERING:

Proposition 5 (OBSERVATION COVERING)
Let $\mathcal{P}$ be a consistent QDP:

1. *Determining whether a given available decision $\vec{d}$ covers a given possible observation $\vec{o}$ is* coNP*-complete*[7]*.*
2. *Determining whether a given possible observation $\vec{o}$ is coverable is Σ_2^p-complete.*

OBSERVATION COVERING still remains coNP-complete under the QID assumption or the STRIPS$^+$ assumption, whatever the choice for $P(\vec{d}, \vec{s})$ (among 1., 2., 3., or 4.), even in the restricted case of unobservability. Moreover, OBSERVATION COVERING remains coNP-complete in the general case under full observability whatever the choice for $P(\vec{d}, \vec{s})$. Additionally, OBSERVATION COVERING remains coNP-complete under the QID or the STRIPS$^+$) assumption in the restricted case of full observability when the choice for $P(\vec{d}, \vec{s})$ is 1. Lastly, in the case of full observability under the QID or the STRIPS$^+$) assumption, and when thc choice for $P(\vec{d}, \vec{s})$ is 2,3 or 4; under the same conditions, the membership to P holds for the choice 1 if G is under conjunctive normal form.

[7]We have also identified the complexity of MPMA as an extra-result: while computing $DepVar(\alpha)$ is already NP-hard, determining if $KB \star^{DepVar(\alpha)} \alpha \models \psi$ holds (given KB, α and ψ) is "only" coNP-complete.

Interestingly, in some cases the notion of cover can be characterized using prime implicants, taking advantage of the connection between 2-QBF, 2-$\overline{\text{QBF}}$ problems and prime implicants given in Proposition 1. Before, we need to rewrite the MPMA into classical deduction (in the same spirit as Lemma 1 in [24] for dependence-based update [16] [17]). The idea consists in (i) duplicating state variables: to each state variable $v \in StateVar$ we associate two state variables v_t and v_{t+1} which refer respectively the the state of the world *before* and *after* the update; note that decision variables need not being duplicated; (ii) translating the effect rules $\langle \varphi, \delta, \psi \rangle$ following this intuitive formulation: if the precondition φ is true at time t (before the update) and if the decision taken satisfies the firing condition δ, then the postcondition ψ is true at time $t+1$ (after the update); (iii) write a *completion* (frame axiom) ensuring that variables not concerned by the update persist. However, in the general case, step (iii) is problematic, because the variables that are not concerned with an update *depend both on the decision taken and on the initial state*, and there are exponentially many possible initial states and available decisions in the worst case. In order to cope with this problem, we proceed the other way round: each state variable is considered separately, and we determine a sufficient condition under which it is not concerned by the update. Of course, this completion depends crucially on the choice made for $P(\vec{d}, \vec{s})$. We will see that if the choice for $P(\vec{d}, \vec{s})$ is 1., 2. or 4., then the sufficient condition found is necessary; only for the case of choice 3., this sufficient condition is not necessary.

For any formula F of $PROP_{PS}$, F_t is obtained from F by replacing each state variable v occurring in it by v_t, and similarly for F_{t+1}. For a world $\vec{x}$, we note $\vec{x}_t$ and $\vec{x}_{t+1}$ the corresponding worlds after time indexing.

Definition 10 (persistence completion) *Let $\mathcal{P}$ be a QDP. We define successively:*

- *for any effect rule $R = \langle \varphi, \delta, \psi \rangle \in EFF$, the translation of R is $Trans(R) = (\varphi_t \wedge \delta) \Rightarrow \psi_{t+1}$;*
- *for any state variable $v \in StateVar$, the* potentially dependent effect rules *of v is defined by:* $PDR(v) = \{\langle \varphi, \delta, \psi \rangle \in EFF \mid v \in DepVar(\psi)\}$;
- *for any state variable $v \in StateVar$, the* persistence completion *of v is defined by:* $Comp(v)$ $= [\bigwedge_{\langle \varphi, \delta, \psi \rangle \in PDR(v)} \neg(\varphi_t \wedge \delta)] \Rightarrow (v_t \Leftrightarrow v_{t+1})$;
- *finally, the translation of EFF is defined by:* $Trans(EFF) = (\bigwedge_{R \in EFF} Trans(R)) \wedge (\bigwedge_{v \in StateVar} Comp(v))$.

Proposition 6 (cover and prime implicants) *For any $\vec{d} \in AvDec$ and $\vec{o} \in PossObs$, $\vec{d}$ covers $\vec{o}$ if and only if there is a γ in $PI^{ObsVar_t \cup DecVar}_{K_t \wedge Trans(EFF)}(C \wedge G_{t+1})$ such that $(\vec{d}, \vec{o}_t) \models \gamma$.*

Example 1 (continued) We have

$$\begin{aligned} & PI^{ObsVar_t \cup DecVar}_{K_t \wedge Trans(EFF)}(C \wedge G_{t+1}) \\ = \{ \; & (\neg cold_t \wedge \neg stuffy_t \wedge \neg open), \\ & (\neg stuffy_t \wedge heating \wedge \neg open), \\ & (summer_t \wedge \neg stuffy_t), (summer_t \wedge open) \; \}. \end{aligned}$$

Accordingly, with $\vec{d}'' = \neg heating \wedge \neg open$, we have

$$\begin{aligned} & PI^{ObsVar_t}_{K_t \wedge Trans(EFF) \wedge \vec{d}''}(C \wedge G_{t+1}) \\ = \; & \{(\neg cold_t \wedge \neg stuffy_t), (summer_t \wedge \neg stuffy_t)\} \end{aligned} .$$

We can check that $\vec{d}''$ covers both $\neg cold \wedge \neg stuffy$ and $summer \wedge \neg stuffy$.

Note that if we replace $DepVar(\psi)$ in the definition of $PDR(v)$ by $Var(\psi)$, then we get a similar result *for choice 2.*. And lastly, if we simply ignore persistence, i.e., if we set $Comp(v) = \emptyset$ for all $v \in StateVar$, then we get a similar result *for choice 1.* No similar result can be obtained for choice 3. Accordingly, the previous proposition shows that if the choice for $P(\vec{d}, \vec{s})$ is 1., 2. or 4., then deciding whether any observation $\vec{o}$ is covered by a given decision $\vec{d}$ is easy, once $PI^{ObsVar_t \cup DecVar}_{K_t \wedge Trans(EFF)}(C \wedge G_{t+1})$ has been computed (this set can be viewed as a compilation for the OBSERVATION COVERING problem). It also shows that OBSERVATION COVERING can be interpreted as an abduction problem: $\vec{d}$ covers $\vec{o}$ iff $\vec{o}_t$ implies a minimal (for $\subseteq$) abductive explanation for $C \wedge G_{t+1}$ w.r.t. $K_t \wedge Trans(EFF) \wedge \vec{d}$, where the set of possible individual hypotheses is $ObsVar_t$ (see [13] for more details).

4.2 Policies

A *policy* for a QDP associates available decisions to possible observations. The resolution of a QDP consists ideally in finding out a policy π mapping every possible observation $\vec{o}$ to an available decision $\vec{d} = \pi(\vec{o})$ that covers it. In this situation, for any possible $\vec{o}$, applying $\pi(\vec{o})$ ensures that a goal state will be reached. Such a policy is called a *solution policy* for $\mathcal{P}$.

Definition 11 (policies)
A partial policy *π for a QDP $\mathcal{P}$ is a mapping from $Dom(\pi) \subseteq PossObs(\mathcal{P})$ to $AvDec(\mathcal{P})$.*
It is sound *iff $\forall \vec{o} \in Dom(\pi)$, $\pi(\vec{o})$ covers $\vec{o}$.*
It is complete *iff $Dom(\pi) = PossObs(\mathcal{P})$.*
A solution policy *is a sound and complete policy.*

Specifying a policy explicitely (by listing all observations of $Dom(\pi)$ together with their assigned decision) is exponentially long in the general case – except in the case where $ObsVar$ has a bounded cardinality – since there may be up to $2^{|ObsVar|}$ observations. So as to deal with this problem, we consider (like in [2]) structured descriptions of policies where decisions are assigned not to complete observations but to (typically partial) observations representing arbitrarily large sets of complete observations.

Definition 12 (structured descriptions)
A structured description of a partial policy is a finite set of pairs $\sigma = \{\langle \vec{o}_i, \vec{d}_i \rangle, i = 1..r\}$.

$\vec{o}_i$ is the *application context* of $\vec{d}_i$. The partial policy π induced by σ is defined in a natural way [8]. Let $\vec{o}$ be a complete observation, then:

if there is a $\langle \vec{o}_i, \vec{d}_i \rangle$ in σ such that $\vec{o} \models \vec{o}_i$
then $\pi(\vec{o}) := \vec{d}_i$ where i is the *first* index of $1 \, .. \, r$ such that $\vec{o} \models \vec{o}_i$
else $\pi(\vec{o})$ is undefined.

Note that we have $Dom(\pi) = \bigvee_{i=1}^{r} \vec{o}_i$.

Clearly enough, OBSERVATION COVERING from a QDP given a structured description σ of a policy for it is in P: it is sufficient to successively consider the application contexts of σ and to check whether one of them is s.t. $\vec{o} \models \vec{o}_i$. If so, $\vec{o}$ is coverable and $\vec{d}_i$ is a covering decision, otherwise $\vec{o}$ is not coverable.

Since it is often the case that there is no solution policy (because some observation is not coverable), we consider now the weaker notion of *maximal sound policy*, i.e., a policy in which every coverable observation is covered (clearly, every solution policy is a maximal sound policy).

Definition 13 (maximal sound policies)
A maximal sound policy π *is a sound policy such that there is no sound policy* π' *that satisfies* $Dom(\pi') \supset Dom(\pi)$.

Example 1 (continued)
With $P(\vec{d}, \vec{s}) = DepVar(Collect(\vec{d}, \vec{s}))$, there is no solution policy for $\mathcal{P}$, because the complete observations (cold $\wedge\neg$summer $\wedge$stuffy) and ($\neg$cold $\wedge\neg$summer $\wedge$stuffy) cannot be covered. Of course, every partial observation that is extracted from any of these two complete observations is not coverable as well. Here is a maximal sound policy under structured form: $\{\langle\neg$summer $\wedge\neg$stuffy, heating $\wedge\neg$open$\rangle$, $\langle$summer, heating $\wedge$open $\rangle\}$. Another maximal sound policy under structured form is: $\{\langle\neg$stuffy, heating $\wedge\neg$open$\rangle$, $\langle$summer $\wedge$stuffy, heating $\wedge$open $\rangle\}$.

We now study the complexity of QDP POLICY EXISTENCE that is the decision problem consisting in determining whether a solution policy exists for a QDP; we make the following parameters vary:

1. observability (partial, full or none);
2. particular assumptions on K, C and EFF.

The following results hold whatever the choice for the parameter $P(\vec{d}, \vec{s})$ (among 1., 2., 3. or 4.).

Proposition 7 (QDP POLICY EXISTENCE)
The complexity of QDP POLICY EXISTENCE *is reported in the following two tables.*

	partially observable
general case	Π_3^p-complete
QID	Π_3^p-complete
STRIPS+	Π_3^p-complete

	fully observable	unobservable
general case	Π_3^p-complete	Σ_2^p-complete
QID	Π_2^p-complete	Σ_2^p-complete
STRIPS+	Π_2^p-complete	Σ_2^p-complete
STRIPS	in P	-

Clearly enough, under unobservability, a solution policy exists iff the empty observation is coverable (i.e., we can find a unique decision that, whatever the initial state, enables to reach a goal state if taken).

It can be shown that solving a QDP amounts at "solving" a quantified boolean formula. Since the formulation is heavy in the general case, we show it in the specific case where $P(\vec{d}, \vec{s}) = PS$. Let $E = \{(\varphi_t \wedge \delta) \Rightarrow \psi_{t+1} \mid \langle \varphi, \delta, \psi \rangle \in EFF\}$. Then $\mathcal{P}$ has a solution policy iff

$$\forall \vec{o}_t \in O_t \; \exists \vec{d} \in D \; \forall \vec{s}_t \models \vec{o}_t \; \forall \vec{s}_{t+1} \in S_{t+1}, \; (\vec{s}_t, \vec{d}, \vec{s}_{t+1}) \models (K_t \wedge E) \Rightarrow (C \wedge G_{t+1})$$

We recognize here an instance of 3-$\overline{\text{QBF}}$. In the particular case of unobservability, it reduces to a 2-QBF problem.

4.3 Computing policies

Computing a policy can be viewed as a form of *compilation* for OBSERVATION COVERING: decisions are associated to observations during an off-line compilation

[8] Since the $\vec{o}_i$ are not required to be mutually inconsistent, it may be the case that more than one $\vec{o}_i$ is satisfied by $\vec{o}$, which means that more than one of the $\vec{d}_i$ covers $\vec{o}$: in this case, any of these decisions covering $\vec{o}$ can be picked up – here we choose arbitrarily to select the one appearing first in σ.

phase, then when the values of the observable variables are known, finding out a decision covering the observation can be achieved on-line in a more efficient way than it would be the case, would this decision be computed from scratch. The time spent in computing a policy can be compensated by the time gained when the same qualitative decision problem is to be frequently repeated with different input observations.

In the following, we present a method for computing structured descriptions of maximal sound policies, based on the computation of prime implicants. Decisions are considered in a successive way, and for each selected decision, an application context is computed. Application contexts are computed for successive decisions $\vec{d}$, and the negation of each of these applications contexts is then added to K so as not to consider the context again.

Input $\mathcal{P}$ (a QDP)
Output σ (a structured description of a partial policy)
Begin
  $\sigma := \emptyset$;
  While $K \wedge G \wedge C$ is consistent **do**
    i. find $\vec{d}$ not appearing in σ and such that $K \wedge G \wedge C \wedge \vec{d}$ is consistent;
    ii. $L := PI^{ObsVar_t}_{K_t \wedge Trans(EFF) \wedge \vec{d}}(C \wedge G_{t+1})$;
    iii. **if** L is not empty **then**
      for each γ_t in L **do**
        add $\langle \gamma, \vec{d} \rangle$ to σ;
        $K \leftarrow K \wedge \neg\gamma$;
      end for
    end if
  end while
  Return σ
End.

At line (i), the search for $\vec{d}$ can be done by searching first for a model of $K \wedge G \wedge C$ and then projecting it on $DecVar$.

At line (ii), $PI^{ObsVar_t}_{K_t \wedge Trans(EFF) \wedge \vec{d}}(C \wedge G_{t+1})$ can be computed in an incremental way. In practice, such prime implicants are derived through the generation of the duals, i.e., prime implicates modulo a formula, for which sophisticated algorithms exist (e.g., [19]). Such algorithms are oriented towards the generation of prime implicates built up from a given subset of propositional variables (here, $ObsVar_t$); they do not require all the prime implicates to be generated, and are typically much more efficient than the naive filtering process that the definition of prime implicants/cates modulo a formula suggests (see [28] for more details).

At line (iii), we first check whether the selected decision covers or not some observations.

Algorithm 1 is correct w.r.t. its specification and has an anytime flavor.

Proposition 8

1. *At any step of the algorithm, σ is a sound partial policy;*
2. *If we let it run quietly, the algorithm stops* [9] *and returns a maximal sound policy.*

5 Related work and conclusion

Our main contribution is a purely propositional framework for qualitative one-stage decision making under uncertainty. In this framework, the existence of a solution policy is closely related to instances of quantified boolean formulas. We have analyzed it from the point of view of computational complexity, and pointed out an algorithm for computing solution policies (exactly if possible, approximately otherwise). It is also the first approach (as far as we know) which makes use of belief update in a decision-theoretic framework. As to planning using updates, the only approach we are aware of is the one by Brewka and Hertzberg [4] who encode the effects of actions by a disjunctive list of transitions $\langle precondition, postcondition \rangle$ and then minimize the set of violated transitions.

Now, the only approach we know that encodes planning problem into QBF is Rintanen's extension of the planning as satisfiability framework ([20]) to conditional planning [30]. His approach shares a lot with ours. Namely, his representation framework corresponds more or less to our STRIPS$^+$ assumption plus atomicity, and the update operator implicitly used for computing the consequences of actions corresponds to the MPMA with any of our choices 2. , 3. or 4. (which all coincide since one action only is performed at each step and effects are terms). However, he considers a multistage framework while ours is restricted to a single stage framework. At this point, it is worth noticing that our framework can be easily extended to fixed-horizon multistage decision making provided that *unobservability is assumed except possibly for the first stage*. With this assumption, we can represent a fixed-horizon planning problem by duplicating state variables and decision variables as many times as necessary as in SATPLAN; then, our policies map initial observations to unconditional sequences of actions.

[9] Within $\min(|AvDec(\mathcal{P})|, |PossObs(\mathcal{P})|)$ loops in the worst case.

As to the practical resolution of QBF, Cadoli et al. [6] propose algorithms for determining whether a given instance of QBF is true or false. This algorithm has been improved by Rintanen in [31]. Contrarily to us, they do not restrict their study to the second and third level of the polynomial hierarchy, and thus consider k-QBF and k-$\overline{\text{QBF}}$ for an arbitrary k. However, both their aim and their techniques differ a lot from ours, more so because their algorithms determine *whether a given QBF is true or not*[10] while ours aim at solving the *function problem* associated to QBF problems (namely, *computing policies*) which has more practical impact when dealing with decision making under uncertainty. The approach by Majercik and Littman [27, 25] is closer to ours since they encode a contingent probabilistic planning problem by a stochastic satisfiability problem (SSAT) which is close to a QBF problem (with "random" quantifiers instead of universal quantifiers). Technically, the resolution of the problem differs from ours (it does use prime implicants).

As to the logical representation of decision processes, the approach by [2] builds structured policies for fully observable, infinite-horizon, stationary Markov Decision Processes. Their algorithm, a structured version of modified policy iteration, incrementally splits application contexts so as to make the current policy converge towards an optimal one. The differences with our approach bear on the assumptions about the process (numerical data, infinite horizon, use of a discounting factor) and on the computational tools: their algorithm consists in iterately updating an "unsound", complete policy by splitting adequate application contexts, and does not make use of prime implicants. This kind of approach is extended to handling partial observability in [3].

We only know two approaches relating planning under incomplete information to abduction. In the first one [26], prime implicants are used to determine the conditions under which a plan always succeeds, may suceed, or always fails. This approach shares a lot with ours in what concerns the search for decisions covering an observation. The second one [32, 11] relates qualitative, *unobservable* decision making to abduction and propose a procedure to compute optimal decisions from stratified knowledge bases and goals. Unlike [26], they do not handle observations. Both approaches do not deal with the construction of policies.

The distinction between controllable and uncontrollable propositional variables goes back to [1] who also gives some abductive characterizations of suitable decisions given a knowledge base and a goal. However [1] does not consider observations and thus does not build conditional policies. The computational complexity of controllability and related notions is studied in [23]. [14] extend the classical CSP framework so as to solve problems of decision under uncertainty (and full observability). While their formalization has some similarities with ours, their algorithm requires strong restrictions on the syntax of the constraints involved (and is thus far less general than ours).

"Supermodels" [15] capture an alternative view of reactive decision under uncertainty: one looks for (unconditional) solutions which are guaranteed to be sufficiently close to a new solution if the data of the problem are slightly modified ("robustness"). Unsurprisingly, finding a supermodel has a lower complexity than finding a solution policy. This approach is based on a notion of distance which is completely absent from our framework (changes from one possible observation to another, or from one decision to another in a policy, are not required to be small); this makes their approach complementary to ours.

[10] Thus, in our framework, it can be used for checking the *existence* of a solution policy.

Acknowledgements

We thank Andreas Herzig and Omar Rifi for helpful discussions about update. The third author has been partly supported by a "Contrat d'objectifs de la Région Nord/Pas-de-Calais" and by the IUT de Lens.

References

[1] C. Boutilier. Toward a logic for qualitative decision theory. In *Proc. of KR'94*, pages 75–86, Bonn, 1994.

[2] C. Boutilier, R. Dearden, and M. Goldszmidt. Exploting structure in policy construction. In *Proc. of IJCAI'95*, pages 1104–1111, Montreal, 1995.

[3] C. Boutilier and D. Poole. Computing optimal policies for partially observable decision processes using compact representations. In *Proc. of AAAI'96*, pages 1168–1175, Portland (OR), 1996.

[4] G. Brewka and J. Hertzberg. How to do things with worlds: on formalizing actions and plans. *Journal of Logic and Computation*, 3(5):517–532, 1993.

[5] T. Bylander. Complexity results for planning. In *Proc. of IJCAI'91*, pages 274–279, Sydney, 1991.

[6] M. Cadoli, A. Giovanardi, and M. Schaerf. An algorithm to evaluate quantified boolean formulae. In *Proc. of AAAI'98*, pages 262–267, 1998.

[7] M.-O. Cordier and P. Siegel. Prioritized transitions for updates. In *Symbolic and Quantitative Approaches to Reasoning and Uncertainty (Proc. of ECSQARU'95)*, Lectures Notes in Artificial Intelligence 946, pages 142–150, Fribourg, 1995. Springer-Verlag.

[8] A. Darwiche and J. Pearl. Symbolic causal networks. In *Proc. of AAAI'94*, pages 238–244, Seattle (WA), 1994.

[9] R. Dechter and I. Rish. Directional resolution: the Davis-Putnam procedure, revisited. In *Proc. of KR'94*, pages 134–145, Bonn, 1994.

[10] P. Doherty, W. Lukascewicz, and E. Madalinska-Bugaj. The PMA and relativizing change for action update. In *Proc. of KR'98*, pages 259–269, 1998.

[11] D. Dubois, D. Le Berre, H. Prade, and R. Sabbadin. Logical representation and computation of optimal decisions in a qualitative setting. In *Proc. of AAAI'98*, pages 588–593, Madison (WI), 1998.

[12] Th. Eiter and G. Gottlob. On the complexity of propositional knowledge base revision, updates and counterfactual. *Artificial Intelligence*, 1992.

[13] Th. Eiter and G. Gottlob. The complexity of logic-based abduction. *Journal of the Association for Computing Machinery*, 42(1):3–42, 1995.

[14] H. Fargier, J. Lang, and Th. Schiex. Mixed constraint satisfaction: a framework for decision making under incomplete knowledge. In *Proc. of AAAI'96*, pages 175–180, Portland (OR), 1996.

[15] M.L. Ginsberg, A.J. Parkes, and A. Roy. Supermodels and robustness. In *Proc. of AAAI'98*, pages 334–339, Madison (WI), 1998.

[16] A. Herzig. The PMA revisited. In *Proc. of KR'96*, pages 40–50, 1996.

[17] A. Herzig and O. Rifi. Propositional belief base update and minimal change. *Artificial Intelligence*, 1999.

[18] R.A. Howard and J.E. Matheson. Influence diagrams. *The Principles and Applications of Decision Analysis*, 2:720–761, 1984.

[19] K. Inoue. Linear resolution in consequence–finding. *Artificial Intelligence*, 56(2–3):301–353, 1992.

[20] H. Kautz and B. Selman. Planning as satisfiability. In *Proc. of ECAI'92*, pages 359–363, 1992.

[21] J. Lang, P. Liberatore, and P. Marquis. The complexity of propositional independence. In preparation, 1999.

[22] J. Lang and P. Marquis. Complexity results for independence and definability. In *Proc. of KR'98*, pages 356–367, Trento, 1998.

[23] J. Lang and P. Marquis. Two forms of dependence in propositional logic: controllability and definability. In *Proc. of AAAI'98*, pages 268–273, Madison (WI), 1998.

[24] P. Liberatore. The complexity of belief update. In *Proc. of IJCAI'97*, pages 68–73, Nagoya, 1997.

[25] M. Littman. Initial experiments in stochastic satisfiability. In *Proc. of IJCAI'99*, pages 667–672, Stockholm, 1999.

[26] W. Lukascewicz and E Madalińska-Bugaj. Reasoning about plans. In *Proc. of IJCAI'97*, 1997.

[27] S. Majercik and M. Littman. Contingent planning under uncertainty via stochastic satisfiability. In *Proc. of AAAI'99*, pages 549–556, Orlando (FL), 1999.

[28] P. Marquis. *Consequence finding algorithms.* in The Handbook for Uncertain and Defeasible Reasoning, Volume 5 (Algorithms for Uncertain and Defeasible Reasoning). Kluwer Academic Publishers, 2000. To appear.

[29] Ch. H. Papadimitriou. *Computational complexity.* Addison–Wesley, 1994.

[30] J. Rintanen. Constructing conditional plans by a theorem prover. *Journal of Artificial Intelligence Research (JAIR)*, 10:323–352, 1999.

[31] J. Rintanen. Improvements to the evaluation of Quantified Boolean Formulae. In *Proc. of IJCAI'99*, pages 1192–1197, Stockholm, 1999.

[32] R. Sabbadin. Decision as abduction. In *Proc. of ECAI'98*, pages 600–604, Brighton, 1998.

[33] L. Savage. *The foundations of statistics.* Wiley and Sons, 1954.

[34] L.J. Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science*, 3:1–22, 1977.

[35] M. Winslett. Reasoning about action using a possible model approach. In *Proc. of AAAI'88*, pages 89–93, St Paul (MN), 1988.