Since January 2020 Elsevier has created a COVID-19 resource centre with free information in English and Mandarin on the novel coronavirus COVID-19. The COVID-19 resource centre is hosted on Elsevier Connect, the company's public news and information website.

Elsevier hereby grants permission to make all its COVID-19-related research that is available on the COVID-19 resource centre - including this research content - immediately available in PubMed Central and other publicly funded repositories, such as the WHO COVID database with rights for unrestricted research re-use and analyses in any form or by any means with acknowledgement of the original source. These permissions are granted for free by Elsevier for as long as the COVID-19 resource centre remains active.
Management of uncertainty orderings through ASP
Andrea Capotorti a and Andrea Formisano b

 a Dipartimento di Matematica e Informatica, Università di Perugia, Italy
 b Dipartimento di Informatica, Università dell’Aquila, Italy

Abstract
Traditionally, most of the proposed probabilistic models of decision under uncertainty rely on numerical measures and representations. Alternative proposals call for qualitative (non-numerical) treatment of uncertainty, based on preference relations and belief orders. The automation of both numerical and non-numerical frameworks surely represents a preliminary step in the development of inference engines of intelligent agents, expert systems, and decision-support tools.

In this paper we exploit Answer Set Programming to formalize and reason about uncertainty expressed as belief orders. The availability of ASP-solvers supports the design of automated tools to handle such formalizations. Our proposal reveals particularly suitable whenever the domain of discernment is partial. We first illustrate how to automatically “classify”, according to the most well-known uncertainty frameworks, any given qualitative uncertainty assessment. Then, we show how to compute an enlargement of the assessment, to any other new inference target.

Key words: Uncertainty orderings, ASP, partial assessments, general inference

Introduction
Uncertainty orderings are receiving wider and wider attention in AI literature, either as theoretical tools to deal directly with belief management [1,6,8], or inside the more articulated framework of decision-making theory [9–11,13]. This burst of interest translates into a wider application of uncertainty orderings, as qualitative assessments better fit the nature of human judgments, while do not suffer intrinsic difficulties typical of numerical elicitations. As happened for numerical methods,
there have been proposed various uncertainty orderings different from traditional probabilistic ones. The main purpose of all of these proposals is to generalize the "additive" character of (unconditional) probabilities. Clearly, different generalizations are possible and several families of qualitative judgments appeared in literature. All of them share common basic properties, but differentiate on the way of combining pieces of information. By following [5,8,16,17], in [2-4] various preference orderings have been fully classified according to their agreement with the most well-known numerical models; both for "complete" (i.e. defined on well structured and closed supports) and "partial" assessments (i.e. defined only among some of the quantities involved). Such axiomatic classifications revealed a coarser grained differentiation with respect to numerical models: different uncertainty measures share the same qualitative properties in combining pieces of information.

The detected classes have been characterized by identifying specific axioms that the orderings must satisfy. Apart from qualitative probabilities, such axioms are of direct declarative reading, as they involve only logical and preference relations. As we will see, such a declarative character supports a straightforward translation of the axioms within the logical framework of ASP. Consequently, we immediately obtain an executable ASP-specification able to discriminate between the different uncertainty orderings. More specifically, this is done by exploiting an ASP-solver (in our case SModels, cf. [18]) that determines which axioms are violated by the given preference relation expressing user's believes comparisons.

Notice that, by proceeding in this way, we actually invert the usual attitude towards qualitative management of uncertainty. In fact, specific axioms are usually set in advance, so that only relations satisfying them are admitted. Here, on the contrary, given a fixed preference relation, our goal consists in ascertain which are the reasonable rules to work with.

In this paper, we mainly focus on the treatment of partial orderings, even if also total relations can be dealt with by the very same machinery. The rational behind this choice is that incomplete assessments appear more realistic models of phenomena which are open to include new (and maybe unexpected) evidences or considerations. In this frame of mind, an interesting problem is that of finding an extension of a preference relation so as to take into account any further event extraneous to the initial assessment. This should be achieved in a way that the initial ordering and its extension satisfy the same axioms. Hence, first, the initial (partial) preference relation is classified by identifying its characterizing rules. Then, by exploiting the same approach the extension sought for is determined.

1. Axioms characterizing partial uncertainty orderings

In [2-4] axioms have been introduced to distinguish alternative possible approaches to qualitative uncertainty. In particular, according to the major uncertainty frameworks, a full classification of partial orderings has been proposed in [4]. In what follows we face a problem left open in [4]. Namely, we propose an inferential
system to classify a given partial ordinal relation into one of the classes.

Let us start by briefly recalling the basic notions on uncertainty orderings and on their axiomatic classification. We will not enter into the details of the motivations for such classification, the reader is referred to [2-4]. The domain of discernment will be represented by a finite set of events \( \mathcal{E} = \{E_1, \ldots, E_n\} \). Among them, \( \phi \) and \( \Omega \) will denote the impossible and the sure event, respectively. The events in \( \mathcal{E} \) are seen as the relevant propositions on which the subject of the analysis can, or wants, to express his/her opinion. Hence, usually \( \mathcal{E} \) does not represent a full model, i.e. it does not comprehend all elementary situations and all of their combinations. For this reason, a crucial component of partial assessments is the knowledge of the logical relationships holding among the events \( E_i \)'s. Usual relationships are those of incompatibility, implication, combination and equivalence. Anyhow, any logical constraint can be potentially expressed. Such constraints are usually represented as a set \( \mathcal{C} \) of clauses among the \( E_i \)'s and such clauses are intended to be always verified. Taking into account \( \mathcal{C} \), the family \( \mathcal{E} \) spans a minimal Boolean algebra \( \mathcal{A}_\mathcal{E} \) containing \( \mathcal{E} \) itself. Note that \( \mathcal{A}_\mathcal{E} \) is only implicitly defined via \( \mathcal{E} \) and \( \mathcal{C} \) and it is not a part of the assessment. Anyway, \( \mathcal{A}_\mathcal{E} \) will be referenced as a supporting structure.

More formally, we have:

**Definition 1.** Let \( \mathcal{E} \) be a set of events and \( \mathcal{C} \) a collection of constraints on \( \mathcal{E} \). \( \mathcal{A}_\mathcal{E} \) is the minimal algebra \((A, \lor, \land, \neg, \phi, \Omega)\) satisfying \( \mathcal{C} \) and such that \( \mathcal{E} \subseteq A \).

Clearly, such an algebra induces a lattice structure on \( A \). The atoms of \( \mathcal{A}_\mathcal{E} \) are the minimal elements of the (sub-)lattice \( \mathcal{A}_\mathcal{E} \setminus \{\phi\} \). Consequently, each event corresponds to a set of atoms, and \( \mathcal{A}_\mathcal{E} \) is (partially) ordered by set inclusion.

An useful piece of notation: in what follows, given any binary relation \( R \), the writing \( \neg(ARB) \) will mean that the pair \((A,B)\) does not belong to \( R \).

**Definition 2.** Let \( \mathcal{A}_\mathcal{E} \) be an algebra of events. A binary relation \( \preccurlyeq^* \) over \( A \) is a (TOTAL) PREFERENCE ORDER if it satisfies the following conditions:

- \((A1)\) \( \preccurlyeq^* \) is a pre-order, i.e. it is reflexive, transitive, and total;
- \((A2)\) \( \phi \preccurlyeq^* \Omega \) and \( \neg(\Omega \preccurlyeq^* \phi) \);
- \((A3)\) for all events \( A, B, A \subseteq B \rightarrow (A \preccurlyeq^* B) \).

Let \( \preccurlyeq^* \) be a total preference order. Then, \( \sim^* \) is its SYMMETRICAL PART, i.e. \( \forall E_1, E_2 (E_1 \sim^* E_2 \leftrightarrow E_1 \preccurlyeq^* E_2 \land E_2 \preccurlyeq^* E_1) \). Moreover, \( \triangleleft^* \) is the ASYMMETRICAL PART of \( \preccurlyeq^* \), i.e. \( \forall E_1, E_2 (E_1 \triangleleft^* E_2 \leftrightarrow E_1 \preccurlyeq^* E_2 \land \neg(E_2 \preccurlyeq^* E_1)) \).

**Definition 3.** Let \( \preccurlyeq \) and \( \triangleleft \) be two binary relations over a set of events \( \mathcal{E} \), such that \( E_1 \triangleleft E_2 \rightarrow E_1 \preccurlyeq E_2 \). The pair \((\preccurlyeq, \triangleleft)\) is a WEAK PREFERENCE STRUCTURE for \( \mathcal{E} \) (w.p.s., for short) if exists a total preference order \( \preccurlyeq^* \) over \( \mathcal{A}_\mathcal{E} \) such that:

\[
\forall E_1, E_2 \in \mathcal{E} \left( (E_1 \preccurlyeq E_2 \rightarrow E_1 \preccurlyeq^* E_2) \land (E_1 \triangleleft E_2 \rightarrow E_1 \triangleleft^* E_2) \right).
\]

Notice that Definition 3 does not require neither \( \preccurlyeq \) or \( \triangleleft \) to be total orders, nor \( \triangleleft \)
to be the asymmetrical part of \(\preceq\). On the other hand, it is required that \(\preceq^*\) extends \(\preceq\), and that \(\prec^*\) (the asymmetrical part of \(\preceq^*\)) extends \(\prec\).

For any weak preference structure \((\preceq, \prec)\) for \(\mathcal{E}\), the following properties hold:

\[(A1')\] if there exist \(E_1, \ldots, E_n \in \mathcal{E}\) such that \(E_1 \preceq E_2 \preceq \cdots \preceq E_n \preceq E_1\), then 
\[
\neg(E_i \prec E_j) \quad \text{for all } i, j \in \{1, \ldots, n\};
\]

\[(A2')\] \(\neg(\Omega \prec \emptyset)\);

\[(A3')\] for all \(E_1, E_2 \in \mathcal{E}\), \(E_1 \prec E_2 \Rightarrow E_2 \not\subseteq E_1\).

Conditions \((A1')-(A3')\) ensure the existence of a total preference order \(\preceq^*\) which enlarges \((\preceq, \prec)\). As a matter of fact, Capacities constitute the most general numerical tool to manage uncertainty, as they express “common sense” behaviors. In our context, any reasonable relation \(\preceq\) must be representable by a partial Capacity (i.e., a restriction of a Capacity measure to the set of events at hand). This corresponds exactly to the satisfaction of the conditions \((A1')-(A3')\). Mathematical properties of orders satisfying such basic axioms are deeply investigated in [6].

**Definition 4.** Let \(\mathcal{E}\) be a set of events. A total preference order \(\preceq^*\) over \(\mathcal{A}_\mathcal{E}\) is said to be **representable** by a numerical measure \(f : \mathcal{A}_\mathcal{E} \rightarrow [0,1]\) iff for all \(E_1, E_2 \in \mathcal{A}_\mathcal{E}\) it holds that \(E_1 \preceq^* E_2 \iff f(E_1) \leq f(E_2)\).

A \w.p.s. \((\preceq, \prec)\) for \(\mathcal{E}\) is said to be **representable** by a partial uncertainty measure \(g : \mathcal{E} \rightarrow [0,1]\) iff it admits an enlargement \(\preceq^*\) over \(\mathcal{A}_\mathcal{E}\) which is representable by an uncertainty measure \(g^* : \mathcal{A}_\mathcal{E} \rightarrow [0,1]\) extension of \(g\) to \(\mathcal{A}_\mathcal{E}\).

Differentiation among ordinal relations can be done on the basis of more specific way of combining different pieces of information. Below, we list the axioms characterizing each class.

**Comparative probabilities.** A \w.p.s. \((\preceq, \prec)\) for \(\mathcal{E}\) can be extended to a total preference order \(\preceq^*\) over \(\mathcal{A}_\mathcal{E}\) representable by a probability function iff (CP) for any \(A_1, \ldots, A_n, B_1, \ldots, B_n \in \mathcal{E}\), with \(B_i \preceq A_i, \forall i = 1, \ldots, n\), such that for some \(r_1, \ldots, r_n > 0\) \(\sup \sum^n_{i=1} r_i (a_i - b_i) \leq 0\) implies that \(A_i \sim B_i\), for all \(i = 1, \ldots, n\) (\(a_i, b_i\) denote the indicator functions of \(A_i, B_i\), resp.).

**Comparative believes.** A \w.p.s. \((\preceq, \prec)\) for \(\mathcal{E}\) can be extended to a total preference order \(\preceq^*\) over \(\mathcal{A}_\mathcal{E}\) representable by a belief function or by a \(n\)-monotone function (with \(n \geq 2\)) iff for all \(A, B, C, D \in \mathcal{E}\) s.t. \(A \subseteq B, C \subseteq D \subseteq B, D \setminus C \subseteq B \setminus A\) it holds

\[(B')\] \(A \sim B \Rightarrow -(C \prec D)\).

**Comparative lower-probabilities.** A \w.p.s. \((\preceq, \prec)\) for \(\mathcal{E}\) can be extended to a total preference order \(\preceq^*\) over \(\mathcal{A}_\mathcal{E}\) representable by a lower-probability function or by a \(0\)-monotone function\(^2\) iff for all \(A, B, C \in \mathcal{E}\) s.t. \(A \subseteq B, C \subseteq B \setminus A\) it holds

\[2\] “\(0\)-monotone” functions are “super-additive” i.e. \(H \wedge K = \phi \rightarrow f(H \vee K) \geq f(H) + f(K)\).

\[1\] We introduce a single axiom for each class, while [4] uses two axioms, since in our characterization we close \(\preceq\) by monotonicity (i.e. \(A \subseteq B \Rightarrow A \preceq B\)).

The name of the classes comes from the representability of \(\preceq^*\) by corresponding numerical measures (recall that qualitative classes are coarser than numerical measures). Axiom (CP) was introduced in [5], while (B’) derives from the analogous axiom introduced for total orders in [17].
Management of uncertainty orderings through ASP

Comparative plausibilities. A w.p.s. \( \prec, \preceq \) for \( \mathcal{E} \) can be extended to a total preference order \( \preceq^* \) over \( \mathcal{A}_E \) representable by a plausibility function or by a \( n \)-alternating function (with \( n \geq 2 \)) iff for all \( A,B,C,D \in \mathcal{E} \) s.t. \( A \subset B, C \subset D \subset B, D \setminus C = B \setminus A \) it holds

\[(PL') \quad A \prec B \rightarrow \neg(C \sim D).
\]

Comparative upper-probabilities. A w.p.s. \( \preceq, \prec \) for \( \mathcal{E} \) can be extended to a total preference order \( \preceq^* \) over \( \mathcal{A}_E \) representable by an upper-probability function or by a \( 0 \)-alternating function\(^3\) iff for all \( A,B \in \mathcal{E} \) s.t. \( A \subset B \), it holds

\[(U') \quad A \prec B \rightarrow \neg(0 \sim B \setminus A).
\]

Comparative lower/upper-probabilities. A w.p.s. \( \preceq, \prec \) for \( \mathcal{E} \) can be extended to a total preference order \( \preceq^* \) over \( \mathcal{A}_E \) representable by both a lower-probability function and an upper-probability function iff it satisfies both \( (L') \) and \( (U') \).

Note that only \( (CP) \) does not have a pure qualitative nature since it involves also indicator functions and summations. It is the only one that should require an external numerical elaboration (e.g. involving some linear programming tool like the simplex or the interior point methods). We also considered the following necessary axiom (i.e. it alone does not guarantee the representability of \( \preceq^* \) by a probability function, but its failure ensure the not representability):

**Weak comparative probabilities.** If \( \prec, \preceq \) can be extended to a total preference order \( \preceq^* \) over \( \mathcal{A}_E \) representable by a probability function then

\[(WC) \quad A \preceq B \rightarrow \neg(B \vee C \prec A \vee C) \text{ for all } A, B, C \in \mathcal{E} \text{ s.t. } A \wedge C = B \wedge C = \phi
\]

2. Answer set programming

In the following sections we show how to obtain executable specifications from the axiomatic classification of preference orders described so far. To this end, we employ Answer Set Programming (ASP, for short).

Let us first briefly recall the basics of such alternative style of logic programming [15]. A problem can be encoded—by using a function-free logic language—as a set of properties and constraints which describe the (candidate) solutions. More specifically, an ASP-program is a collection of rules of the form

\[L_1; \ldots; L_k; \neg L_{k+1}; \ldots; \neg L_\ell \leftarrow L_{\ell+1}, \ldots, L_m, \neg L_{m+1}, \ldots, \neg L_n \]

where \( n \geq m \geq \ell \geq k \geq 0 \) and each \( L_i \) is a literal, i.e., an atom \( A \) or a negation of an atom \( \neg A \). The symbol \( \neg \) denotes classical negation, while \( \neg \) stands for negation-as-failure (Notice that \( ', ' \) and \( ', ' \) stand for logical conjunction and disjunction, respectively.) The left-hand side and the right-hand side of the clause are said head and body, respectively. A rule with empty head is a constraint. The literals in the body of a constraint cannot be all true, otherwise they would imply falsity.

Semantics of ASP is expressed in terms of answer sets (or equivalently stable models, cf. [12]). Consider first the case of an ASP-program \( P \) which does not

\(^3\) “0-alternating” functions are “sub-additive” i.e. \( H \wedge K = \phi \rightarrow f(H \vee K) \leq f(H) + f(K).\)
involve negation-as-failure (i.e., \( \ell = k \) and \( n = m \)). In this case, a set \( X \) of literals is said to be closed under \( P \) if for each rule in \( P \), whenever \( \{L_{\ell+1}, \ldots, L_m\} \subseteq X \), it holds that \( \{L_1, \ldots, L_k\} \cap X \neq \emptyset \). If \( X \) is inclusion-minimal among the sets closed under \( P \), then it is said to be an answer set for \( P \). Such a definition is extended to any program \( P \) containing negation-as-failure by considering the reduct \( P^X \) (of \( P \)). \( P^X \) is defined as the set of rules of the form \( L_1; \ldots; L_k \leftarrow L_{\ell+1}, \ldots, L_m \) for all rules of \( P \) such that \( X \) contains all the literals \( L_{k+1}, \ldots, L_t \), but does not contain any of the literals \( L_{m+1}, \ldots, L_n \). Clearly, \( P^X \) does not involve negation-as-failure. The set \( X \) is an answer set for \( P \) if it is an answer set for \( P^X \).

Once a problem is described as an ASP-program \( P \), its solutions (if any) are represented by the answer sets of \( P \). Notice that an ASP-program may have none, one, or several answer sets.

Let us consider the program \( P \) consisting of the two rules \( p; q \leftarrow \) and \( \neg r \leftarrow p \). Such a program has two answer sets: \{\( p, \neg r \)\} and \{\( q \)\}. If we add the rule (actually, a constraint) \( \leftarrow q \) to \( P \), then we rule-out the second of these answer sets, because it violates the new constraint. This simple example reveals the core of the usual approach followed in formalizing/solving a problem with ASP. Intuitively speaking, the programmer adopts a “generate-and-test” strategy: first (s)he provides a set of rules describing the collection of (all) potential solutions. Then, the addition of a group of constraints rules-out all those answer sets that are not desired real solutions.

To find the solutions of an ASP-program, an ASP-solver is used. Several solvers have become available (cf. [18], for instance), each of them being characterized by its own prominent valuable features. In this work we choose smodels as solver, together with its natural front-end lparse [14].

Expressive power of ASP, as well as, its computational complexity have been deeply investigated. The interested reader can refer to the survey [7], among others.

Let us give a simple example of ASP-program. In doing this, we will recall the syntax and the main features of lparse/smodels which will be exploited in the rest of the paper (see [14], for a more details). The problem we want to formalize in ASP is the well-known \( n \)-queens problem: “Given a \( n \times n \) chess board, place \( n \) queens in such a way that no two of them attack each other”. The two clauses below state that a candidate solution is any disposition of the queens, provided that each column of the board contains one and only one queen. (The fact that a queen is placed on the \( n^{th} \) column and on the \( m^{th} \) row is encoded by the atom \( \text{queen}(n,m).) \)

\[
\text{pos}(1..n). \quad 1\{\text{queen}(\text{Col},\text{Row}) : \text{pos}(	ext{Col})\}1 : - \text{pos}(	ext{Row}).
\]

The second rule is a particular form of constraint available in smodels’ language. The general form of such a kind of clauses is

\[
k\{\langle \text{property.def} \rangle; \langle \text{range.def} \rangle \}m : -(\text{search.space})
\]

where: the conditions \( \langle \text{search.space} \rangle \) in the body define the set of objects of the domain to be checked; the atom \( \langle \text{property.def} \rangle \) in the head defines the property to be

\footnote{In the syntax of smodels ‘-\( m \)’ denotes implication \( \leftarrow \). The value of the constant \( n \) occurring in the first clause is a parameter of the program supplied to lparse at run-time.}
checked; the conjunction \((\text{range.def})\) defines the possible values that the property may take on the objects defined in the body, namely by providing a conjunction of unary predicates each of them defining a range for one of the variables that occur in \((\text{property.def})\) but not in \((\text{search_space})\); \(k\) and \(m\) are the minimum and maximum number of values that the specified property may take on the specified objects.

We now introduce two constraints, in order to rule out those placements where two queens control either the same row or the same diagonal of the board:

\[
\begin{align*}
\texttt{:}- \text{queen}(C,R_1), \text{queen}(C,R_2), \text{pos}(C;R_1;R_2), R_1 &< R_2. \\
\texttt{:}- \text{queen}(C_1,R_1), \text{queen}(C_2,R_2), \text{pos}(C_1;C_2;R_1;R_2), R_1 &< R_2, \text{abs}(C_1-C_2)==\text{abs}(R_1-R_2).
\end{align*}
\]

Here is one of the answer sets produced by smodels, when fed with our program (in this case we put \(n=8\)): Answer: 1. queen(4,1), queen(6,2), queen(1,3), queen(5,4), queen(2,5), queen(8,6), queen(3,7), queen(7,8).

Notice that \texttt{lparse} offers some elementary built-in arithmetic functions (such as \texttt{abs()}, in the above clause) that can be used to perform simple arithmetics. More in general, \texttt{lparse} allows the user to employ user-defined C or \texttt{C++} functions within an ASP-program. The object code of these functions is linked with \texttt{lparse} at runtime (cf. [14]). We exploited this feature (not available in some other solvers) to implement a basic library of functions aimed at handling sets and operation on sets.

The pair \texttt{lparse}/\texttt{smodels} constitutes an essential and neat tool for fast prototypical development. Notable facilities come from the simple albeit useful integration capabilities with the C programming language, the prompt availability of the source-code and documentation, and the ease of use.

3. Preference classification

Our first task consists in writing an ASP-program able to classify any given \(w.p.s.\) \((\preceq, \prec)\) w.r.t. the axioms seen in Sec. 1 (Except for \((\text{CP})\), that, up to our knowledge, does not admit a purely declarative formulation). We start by defining in ASP the predicates \texttt{prec}(-,-), \texttt{precneq}(-,-), and \texttt{equiv}(-,-), to render the relations \(\preceq, \prec,\) and \(\sim\), resp. Moreover, the fact of “being an event” (i.e. a member of \(\mathcal{E}\)) is stated through the monadic predicate \texttt{event}(-).\(^6\) Auxiliary predicates/functions are used to render set-theoretical construks, such as \(\cap, \cup, \) and \(\subseteq\), whose implementations rely on user-defined C-libraries. The characterization of potential legal answer sets is done by asserting properties of \texttt{prec}(-,-), \texttt{precneq}(-,-), and \texttt{equiv}(-,-), as follows:

\[
\begin{align*}
\texttt{prec}(A,B) &\text{ :- } \texttt{event}(A;B), \texttt{equiv}(A,B).
\texttt{prec}(B,A) &\text{ :- } \texttt{event}(A;B), \texttt{equiv}(A,B).
\texttt{:}- \texttt{precneq}(A,B), \texttt{event}(A;B), \texttt{equiv}(A,B).
\end{align*}
\]

\(^5\) Here \texttt{pos}(C;R_1;R_2) is a shorthand notation for the three facts \texttt{pos}(C), \texttt{pos}(R_1), \texttt{pos}(R_2).
\(^6\) Actually, in our program, events are denoted by integer numbers. Here, for the sake of readability, we systematically denote events by capital letters.
prec(A,B) :- event(A;B), precneq(A,B).
equiv(A,B) :- event(A;B), prec(B,A), prec(A,B).

Also axioms (A1')–(A3') must be imposed. For instance (A3') is rendered by weeding out all answer sets where \( A \subset B \wedge B \prec A \) holds for some events \( A \) and \( B \):
\[
\text{failsBl} :- \text{event}(A;B;C), \text{subset}(A,B), A \neq B, \text{empty}(B \cap C), \text{precneq}(A,B), \text{prec}(B \cup C, A \cup C).
\]

Having in mind (B') of Sec. 1, this clause is of immediate reading: the fact failsBl is true (i.e. belongs to the answer set) whenever there exist events falsifying (B'). All other axioms have been treated similarly. When smodels is fed with such program and a description of an input preference relation (i.e., a collection of facts of the forms \( \text{prec}(\cdot,\cdot), \text{precneq}(\cdot,\cdot), \) and \( \text{equiv}(\cdot,\cdot) \)), different outcomes may be obtained:

a) If no answer set is produced, then the input weak preference structure violates some basic requirement, such as axioms (A1')–(A3').

b) Otherwise, if an answer set is generated, there exists a numerical (partial) model representing the input weak preference structure. Moreover, the presence in the answer set of a fact of the form failsC (say failsL1, for example), witnesses that the corresponding axiom ((L') in the case) is violated by the given preference order. Consequently, the given order (as well as its extensions) is not compatible with the uncertainty framework ruled by \( C \) (in the case of failsL1, the given order cannot be represented by a partial Lower probability).

**Example 1.** Suppose a physician wants to perform a preliminary evaluation about the reliability of a test for SARS (Severe Acute Respiratory Syndrome). Up to his/her knowledge, the SARS diagnosis is based on moderate or severe respiratory symptoms and on the positivity or indeterminacy of an adopted clinical test about the presence of the SARS-associated antibody coronavirus SARS-CoV). The elements appearing in his/her analysis can be schematized as:

\[
\begin{align*}
A &\equiv \text{Normal respiratory symptoms}, \\
B &\equiv \text{Moderate respiratory symptoms}, \\
C &\equiv \text{Severe respiratory symptoms}, \\
D &\equiv \text{Moderate or severe respiratory symptoms}, \\
E &\equiv \text{Death from pulmonary diseases}, \\
F &\equiv \text{Positive or indeterminate clinical test},
\end{align*}
\]

subject to these (logical) restrictions: \( A \cap B = \emptyset, B \cap C = \emptyset, A \cap C = \emptyset, A \cup B \cup C = \Omega, D = B \cup C, E \subset C, F \cap A = \emptyset \). Consider the w.p.s. \((\preceq, \prec)\) so described: \( \text{precneq}(\emptyset,C), \text{precneq}(C,B), \text{prec}(B,A), \text{precneq}(C,D), \text{precneq}(E,C), \text{precneq}(E,D), \text{precneq}(F,A), \text{equiv}(A \cup E, A \cup C) \).

Due to events’ meaning, such order seems reasonable. If it is given as input to smodels, the answer set found includes the facts failsB1 and failsWC. This means that the given preference relation agrees with the basic axioms, however it cannot be managed by using neither a Probability nor a Belief function. Nevertheless, one can use comparative Lower probabilities or comparative Plausibilities.
4. Partial-order extension

An interesting problem is that of finding an extension of a preference relation so as to take into account any further event extraneous to the initial assessment. This should be achieved in a way that the extension retains the same character of the initial order (e.g., both satisfy the same axioms). More precisely, let be given an initial (partial) assessment expressed as a (partial) order \( \preceq \) over set of known events \( \mathcal{E} \). Assume that \( \preceq \) satisfies the axioms characterizing a specific class, say \( \mathcal{C} \), of orders (cf. Sec. 3). Consider a new event \( S \) (not in \( \mathcal{E} \)), implicitly described by a collection \( \mathcal{C}' \) of set-theoretical constraints involving the events of \( \mathcal{E} \). In the spirit of [5, Thm. 3], the problem we are going to tackle is: Determine which is the “minimal” extension \( \preceq^{+} \) (over \( \mathcal{E} \cup \{S\} \)) of \( \preceq \), induced by the new event, which still belongs to the class \( \mathcal{C} \). In other words, we are interested in ascertaining how the new event \( S \) must relate to the members of \( \mathcal{E} \) in order that \( \preceq^{+} \) still is in \( \mathcal{C} \). To this aim we want to determine the sub-collections \( \mathcal{L}_S \), \( \mathcal{W}_S \), \( \mathcal{U}_S \), and \( \mathcal{W}_U \), of \( \mathcal{E} \) so defined:

- \( E \in \mathcal{L}_S \) iff no extension \( \preceq^* \) of \( \preceq \) can infer that \( S \preceq^* E \)
- \( E \in \mathcal{W}_S \) iff no extension \( \preceq^* \) of \( \preceq \) can infer that \( S \prec^* E \)
- \( E \in \mathcal{U}_S \) iff no extension \( \preceq^* \) of \( \preceq \) can infer that \( E \preceq^* S \)
- \( E \in \mathcal{W}_U \) iff no extension \( \preceq^* \) of \( \preceq \) can infer that \( E \prec^* S \)

Consequently, in order to satisfy the axioms characterizing \( \mathcal{C} \), any weak preference structure \( (\preceq^+, \prec^+) \) extending \( (\preceq, \prec) \) must, at least, impose that:

\[
E \prec^+ S \forall E \in \mathcal{L}_S, \quad E \preceq^+ S \forall E \in \mathcal{W}_S, \quad S \prec^+ E \forall E \in \mathcal{U}_S, \quad S \preceq^+ E \forall E \in \mathcal{W}_U.
\]

We describe now an ASP-program that solves this problem by taking advantage from the computation done during the classification phase: It gets as input the knowledge regarding the satisfied axiom(s), the preference and logical relations on the original set of events, and the description of the new event (see Example 2).

The handling of the axioms is done by ASP-rules of the form (we give here the rule for \( (L') \)). Other axioms are treated similarly):

\[
\text{:- holdsL1, event(A;B), empty(N), empty(A \cap B), precneq(N,A), prec(A \cup B,B).}
\]

Rules of this kind (actually, constraints, in the sense described in Sec. 2), declare “undesirable” any extension for which the axiom is violated. For instance, consider a ground instance of the above rule; whenever the fact holdsL1 is present (i.e. is true in an answer set), then to make the (ground) clause satisfied, at least one of the other literals must not belong to the answer set. (Notice that, these literals are all true exactly when \( (L') \) is violated.) Consequently, in order to activate this constraint (i.e. to impose axiom \( (L') \), for the case at hand) it suffices to add the fact holdsL1 to the input of the solver.

A further rule describes the potential answer set we are interested in:

\[
\text{1{ precneq(A,B), equiv(A,B), precneq(B,A) }}1 \text{ :- event(A;B).}
\]

This rule simply asserts that any computed answer-set must predicate on each pair \( A,B \) of events by stating exactly one, and only one, of the three facts \( \text{precneq}(A,B), \text{equiv}(A,B), \text{and precneq}(B,A) \). Then, smodels calculates the answer sets
fulfilling the desired requirements and encoding "legal" total orders.

The collections $\mathcal{L}_S$, $\mathcal{W}\mathcal{L}_S$, $\mathcal{U}_S$, and $\mathcal{W}\mathcal{U}_S$ can be obtained by computing the intersection $C_n$ of all these answer sets. (Or, equivalently, by computing the set of logical consequences of the ASP-program. Notice that, in general, $C_n$ needs not to be an answer set by itself.) Unfortunately, not all the available ASP-solvers offer the direct computation of $C_n$ as a built-in feature (DLV, for instance does, while smodels does not, cf. [18]). In general, a simple inspection of the answer sets generated by smodels allows one to detect the minimal extension of the preference relation which is mandatory for each total order. We designed a simple post-processor which filters smodels’ output and produces the imposed extension of $\preceq$.

**Example 2.** Consider the weak preference structure of Example 1 and the new event

$S = \text{The real state of having SARS}$

subject to these restrictions: $S \subseteq F$ and $F \cap E \subseteq S$. Since in Example 1 we discovered that the initial preference relation satisfies axiom (PL'), we want to impose such axiom and compute the extension of the initial order. Once filtered smodels’ output, we obtained the following result: 7 $\preceq(S,A\cup C)$ $\preceq(S,A\cup E)$ $\preceq(S,D)$ $\preceq(S,A)$ $\preceq(S,\Omega)$ $\preceq(\emptyset,S)$ $\preceq(S,F)$ ... showing that, apart from obvious relations induced by monotonicity, no significative constraint involving $S$ can be inferred. Since $S$ and $E$ can be freely compared, this result suggests that either further investigation about relevance of the clinical test or a revision of the initial preference relation, should be performed.

The availability of automated tools able to extend preference orders, whenever new knowledge (new events) is acquired, directly suggests applications in expert systems and decision-support tools. In automated diagnosis, planning, or problem solving, to mention some examples, one could easily imagine scenarios where knowledge is not entirely available from the beginning. We could outline how a rudimental inference process could develop, by identifying the basic steps an automated agent should perform:

0) Acquisition of an initial collection of observations (events) about the object of the analysis, together with a (qualitative) partial preference assessment;

1) Detection of which is the most adequate (i.e., the most discriminant) uncertainty framework, through a “preference classification phase” (cf. Sec. 3);

2) Whenever new knowledge becomes available, refine agent’s description of the real world by performing order extension (which substantially corresponds to knowledge inference. Cf. Sec. 4).

The results of step 2) could be then exploited to guide further investigations on the real world, in order to obtain new information. Then, step 2) will be repeated and the process will continue until further pieces of knowledge are obtainable or an enough accurate degree of believe is achieved.

7 We list here only the portion of the extension involving the new event $S$. 
Conclusions

In this paper we started an exploration of the potentialities offered by ASP for building decision support systems based on qualitative judgments. The implementation of what could be thought as a kernel of an inference engine sprouted almost naturally. Certainly, our research is at an initial stage and the implementation we reported on is just a prototype. Next step in this research would consist in validating the proposed approach by means of a number of benchmarks aimed at testing our prototype on the ground of real applications. Results of this activity will help in consolidating the prototype. In this context, a further goal consists in completing our approach so as to handle comparative Probabilities too. Since no axiomatic characterization of comparative Probabilities is known (up to our knowledge), this aim should be achieved through integration with efficient linear optimization tools (such as the column generation techniques). More in general, we envisage the design of a full-blown automated system which integrates different (in someway complementary) techniques and methods for uncertainty management; comprehending mixed numerical/qualitative assessments and conditional frameworks.

References

1. T. Bilgiç (2001) Fusing interval preferences, in Proc. of EUROFUSE Workshop on Preference Modelling and Applications, pp. 253–258.
2. A. Capotorti, G. Coletti, and B. Vantaggi (1998) Non additive ordinal relations representable by lower or upper probabilities. Kybernetika, 34(1):79–90.
3. A. Capotorti and B. Vantaggi (1998) Relationships among ordinal relations on a finite set of events. In Information, Uncertainty, Fusion, Eds. B. Bouchon-Meunier, R. R. Yager and L. A. Zadeh, pp. 447–458.
4. A. Capotorti and B. Vantaggi (2000) Axiomatic characterization of partial ordinal relations. Int. J. of Approximate Reasoning, 24:207–219.
5. G. Coletti (1990) Coherent qualitative probability. J. Math. Psych., 34:297–310.
6. G. de Cooman (1997) Confidence Relations and Ordinal Information. Inform. Sci., 104.
7. E. Dantsin, T. Eiter, G. Gottlob, and A. Voronkov (2001) Complexity and expressive power of logic programming. ACM Comput. Surveys, 33(3):374–425.
8. D. Dubois (1986) Belief Structure, Possibility Theory and Decomposable Confidence Measures on Finite Sets. Comput. Artif. Intell., 5(5):403–416.
9. D. Dubois, H. Fargier, and H. Prade (1997) Decision-making under ordinal preferences and uncertainty, AAAI Spring Symposium on Qualitative Preferences in Deliberation and Practical Reasoning, pp. 41–46.
10. D. Dubois, H. Prade, and R. Sabbadin (1997) A possibilistic logic machinery for qualitative decision, AAAI Spring Symposium on Qualitative Preferences in Deliberation and Practical Reasoning, pp. 47–54.
11. P. H. Giang, P. P. Shenoy (2001) A comparison of axiomatic approaches to qualitative decision making under possibility theory, *Proc. of the 17th Conf. on Uncertainty in Artificial Intelligence UAI01*, pp. 162–170.

12. M. Gelfond and V. Lifschitz (1988) *The stable model semantics for logic programming*, *Proc. ILPS-5*, pp. 1070–1080.

13. D. Lehmann (1996) Generalized qualitative probability: Savage revisited, *Proc. of 12th Conf. on Uncertainty in Artificial Intelligence UAI96*, pp. 381–388.

14. T. Syrjänen (2001) *Lparse 1.0 User’s Manual*.

15. W. Marek and M. Truszczynski (1999) *Stable models and an alternative logic programming paradigm*. The Logic Programming Paradigm: a 25-Year Perspective, Springer, pp. 375–398.

16. P. Walley and T. Fine (1979) Varieties of modal (classificatory) and comparative probability. *Synthese*, 41:321–374.

17. S. K. M. Wong, Y. Y. Yao, P. Bollmann, and H. C. Bürger (1991) Axiomatization of Qualitative Belief Structure. *IEEE Trans. on System, Man, and Cybernetics*, 21:726–734.

18. Web resources for ASP solvers. CModels: [www.cs.utexas.edu/users/tag/cmodels.html](http://www.cs.utexas.edu/users/tag/cmodels.html); DLV: [www.dbai.tuwien.ac.at/proj/dlv](http://www.dbai.tuwien.ac.at/proj/dlv); SModels: [www.tcs.hut.fi/Software/smodels](http://www.tcs.hut.fi/Software/smodels);