A Declarative Approach for Computing Ordinal Conditional Functions Using Constraint Logic Programming

Christoph Beierle[1](✉), Gabriele Kern-Isberner[2], and Karl Södler[1]

[1] Department of Computer Science, FernUniversität in Hagen, 58084 Hagen, Germany
christoph.beierle@fernuni-hagen.de

[2] Department of Computer Science, TU Dortmund, 44221 Dortmund, Germany

Abstract. In order to give appropriate semantics to qualitative conditionals of the form *if A then normally B*, ordinal conditional functions (OCFs) ranking the possible worlds according to their degree of plausibility can be used. An OCF accepting all conditionals of a knowledge base R can be characterized as the solution of a constraint satisfaction problem. We present a high-level, declarative approach using constraint logic programming (CLP) techniques for solving this constraint satisfaction problem. In particular, the approach developed here supports the generation of all minimal solutions; this also holds for different notions of minimality which we discuss and implement in CLP. Minimal solutions are of special interest as they provide a basis for model-based inference from R.

1 Introduction

In knowledge representation, rules play a prominent role. Default rules of the form *If A then normally B* are being investigated in nonmonotonic reasoning, and various semantical approaches have been proposed for such rules. Since it is not possible to assign a simple Boolean truth value to such default rules, a semantical approach is to define when a rational agent accepts such a rule. We could say that an agent accepts the rule *Birds normally fly* if she considers a world with a flying bird to be less surprising than a world with a nonflying bird. At the same time, the agent can also accept the rule *Penguin birds normally do not fly*; this is the case if she considers a world with a nonflying penguin bird to be less surprising than a world with a flying penguin bird.

The informal notions just used can be made precise by formalizing the underlying concepts like default rules, epistemic state of an agent, and the acceptance relation between epistemic states and default rules. In the following, we deal with qualitative default rules and a corresponding semantics modelling the epistemic state of an agent. While a full epistemic state could compare possible worlds

The research reported here was partially supported by the Deutsche Forschungsgemeinschaft – DFG (grants BE 1700/7-2 and KE 1413/2-2).

according to their possibility, their probability, their degree of plausibility, etc. (cf. [9,10,18]), we will use ordinal conditional functions (OCFs), which are also called ranking functions [18]. To each possible world ω, an OCF κ assigns a natural number $\kappa(\omega)$ indicating its degree of surprise: The higher $\kappa(\omega)$, the greater is the surprise for observing ω.

In [12,13] a criterion when a ranking function respects the conditional structure of a set $\mathcal{R}$ of conditionals is defined, leading to the notion of c-representation for $\mathcal{R}$, and it is argued that ranking functions defined by c-representations are of particular interest for model-based inference. In [3] a system that computes a c-representation for any such $\mathcal{R}$ that is consistent is described, but this c-representation may not be minimal. An algorithm for computing a minimal ranking function is given in [5], but this algorithm fails to find all minimal ranking functions if there is more than one minimal one. In [15] an extension of that algorithm being able to compute all minimal c-representations for $\mathcal{R}$ is presented. The algorithm developed in [15] uses a non-declarative approach and is implemented in an imperative programming language. While the problem of specifying all c-representations for $\mathcal{R}$ is formalized as an abstract, problem-oriented constraint satisfaction problem $CR(\mathcal{R})$ in [2], no solving method is given there.

In this paper, we present a high-level, declarative approach using constraint logic programming techniques for solving the constraint satisfaction problem $CR(\mathcal{R})$ for any consistent $\mathcal{R}$. In particular, the approach developed here supports the generation of all minimal solutions; these minimal solutions are of special interest as they provide a preferred basis for model-based inference from $\mathcal{R}$. Moreover, we investigate different notions of minimality and demonstrate the flexibility of our approach by showing how alternative minimality concepts can be taken into account by slight modifications of the CLP implementation.

The rest of this paper is organized as follows: After recalling the formal background of conditional logics as it is given in [1] and as far as it is needed here (Sect. 2), we elaborate the birds-penguins scenario sketched above as an illustration for a conditional knowledge base and its semantics in Sect. 3. The definition of the constraint satisfaction problem $CR(\mathcal{R})$ and its solution set denoting all c-representations for $\mathcal{R}$ is given in Sect. 4. In Sect. 5, a declarative, high-level CLP program `GenOCF` solving $CR(\mathcal{R})$ is developed, observing the objective of being as close as possible to $CR(\mathcal{R})$. Its realization in Prolog is described in detail, as well as the modifications needed for alternative notions of minimality. In Sect. 6, `GenOCF` is evaluated with respect to a series of some first example applications. Section 7 concludes the paper and points out further work.

2 Background

We start with a propositional language $\mathcal{L}$, generated by a finite set Σ of atoms $a, b, c, \ldots$. The formulas of $\mathcal{L}$ will be denoted by uppercase Roman letters $A, B, C, \ldots$. For conciseness of notation, we will omit the logical *and*-connective, writing AB instead of $A \wedge B$, and overlining formulas will indicate negation, i.e.

$\overline{A}$ means $\neg A$. Let Ω denote the set of possible worlds over $\mathcal{L}$; Ω will be taken here simply as the set of all propositional interpretations over $\mathcal{L}$ and can be identified with the set of all complete conjunctions over Σ. For $\omega \in \Omega$, $\omega \models A$ means that the propositional formula $A \in \mathcal{L}$ holds in the possible world ω.

By introducing a new binary operator $|$, we obtain the set $(\mathcal{L} \mid \mathcal{L}) = \{(B|A) \mid A, B \in \mathcal{L}\}$ of *conditionals* over $\mathcal{L}$. $(B|A)$ formalizes "*if A then (normally) B*" and establishes a plausible, probable, possible etc. connection between the *antecedent* A and the *consequence* B. Here, conditionals are supposed not to be nested, that is, antecedent and consequent of a conditional will be propositional formulas.

A conditional $(B|A)$ is an object of a three-valued nature, partitioning the set of worlds Ω in three parts: those worlds satisfying AB, thus *verifying* the conditional, those worlds satisfying $A\overline{B}$, thus *falsifying* the conditional, and those worlds not fulfilling the premise A and so which the conditional may not be applied to at all. This allows us to represent $(B|A)$ as a *generalized indicator function* going back to [7] (where u stands for *unknown* or *indeterminate*):

$$(B|A)(\omega) = \begin{cases} 1 & \text{if } \omega \models AB \\ 0 & \text{if } \omega \models A\overline{B} \\ u & \text{if } \omega \models \overline{A} \end{cases} \tag{1}$$

To give appropriate semantics to conditionals, they are usually considered within richer structures such as *epistemic states*. Besides certain (logical) knowledge, epistemic states also allow the representation of preferences, beliefs, assumptions of an intelligent agent. Basically, an epistemic state allows one to compare formulas or worlds with respect to plausibility, possibility, necessity, probability, etc.

Well-known qualitative, ordinal approaches to represent epistemic states are Spohn's *ordinal conditional functions, OCFs*, (also called *ranking functions*) [18], and *possibility distributions* [4], assigning degrees of plausibility, or of possibility, respectively, to formulas and possible worlds. In such qualitative frameworks, a conditional $(B|A)$ is valid (or *accepted*), if its confirmation, AB, is more plausible, possible, etc. than its refutation, $A\overline{B}$; a suitable degree of acceptance is calculated from the degrees associated with AB and $A\overline{B}$.

In this paper, we consider Spohn's OCFs [18]. An OCF is a function

$$\kappa : \Omega \to \mathbb{N}$$

expressing degrees of plausibility of propositional formulas where a higher degree denotes "less plausible" or "more suprising". At least one world must be regarded as being normal; therefore, $\kappa(\omega) = 0$ for at least one $\omega \in \Omega$. Each such ranking function can be taken as the representation of a full epistemic state of an agent. Each such κ uniquely extends to a function (also denoted by κ) mapping sentences and rules to $\mathbb{N} \cup \{\infty\}$ and being defined by

$$\kappa(A) = \begin{cases} \min\{\kappa(\omega) \mid \omega \models A\} & \text{if } A \text{ is satisfiable} \\ \infty & \text{otherwise} \end{cases} \tag{2}$$

for sentences $A \in \mathcal{L}$ and by

$$\kappa((B|A)) = \begin{cases} \kappa(AB) - \kappa(A) & \text{if } \kappa(A) \neq \infty \\ \infty & \text{otherwise} \end{cases} \tag{3}$$

for conditionals $(B|A) \in (\mathcal{L} \mid \mathcal{L})$. Note that $\kappa((B|A)) \geqslant 0$ since any ω satisfying AB also satisfies A and therefore $\kappa(AB) \geqslant \kappa(A)$.

The belief of an agent being in epistemic state κ with respect to a default rule $(B|A)$ is determined by the satisfaction relation $\models_{\mathcal{O}}$ defined by:

$$\kappa \models_{\mathcal{O}} (B|A) \text{ iff } \kappa(AB) \; < \; \kappa(A\overline{B}) \tag{4}$$

Thus, $(B|A)$ is believed in κ iff the rank of AB (verifying the conditional) is strictly smaller than the rank of $A\overline{B}$ (falsifying the conditional). We say that κ *accepts* the conditional $(B|A)$ iff $\kappa \models_{\mathcal{O}} (B|A)$.

3 Example

In order to illustrate the concepts presented in the previous section, we will use a scenario involving a set of some default rules representing common-sense knowledge.

Example 1. Suppose we have the propositional atoms
f - *flying*, b - *birds*, p - *penguins*, w - *winged* animals, k - *kiwis*.
Let the set $\mathcal{R}$ consist of the following conditionals:

$\mathcal{R}$	r_1:	$(f\|b)$	*birds fly*
	r_2:	$(b\|p)$	*penguins are birds*
	r_3:	$(\overline{f}\|p)$	*penguins do not fly*
	r_4:	$(w\|b)$	*birds have wings*
	r_5:	$(b\|k)$	*kiwis are birds*

Figure 1 shows a ranking function κ that accepts all conditionals given in $\mathcal{R}$. Thus, for any $i \in \{1, 2, 3, 4, 5\}$ it holds that $\kappa \models_{\mathcal{O}} R_i$.

For the conditional $(f|p)$ (*"Do penguins fly?"*) that is not contained in $\mathcal{R}$, we get $\kappa(pf) = 2$and $\kappa(p\overline{f}) = 1$ and therefore

$$\kappa \not\models_{\mathcal{O}} (f|p)$$

so that the conditional $(f|p)$ is not accepted by κ. This is in accordance with the behaviour of a rational agent believing $\mathcal{R}$ since the knowledge base $\mathcal{R}$ used for building up κ explicitly contains the opposite rule $(\overline{f}|p)$.

On the other hand, for the conditional $(w|k)$ (*"Do kiwis have wings?"*) that is also not contained in $\mathcal{R}$, we get $\kappa(kw) = 0$ and $\kappa(k\overline{w}) = 1$ and therefore

$$\kappa \models_{\mathcal{O}} (w|k)$$

i.e., the conditional $(w|k)$ is accepted by κ. Thus, from their superclass *birds*, kiwis inherit the property of having wings.

ω	$\kappa(\omega)$	ω	$\kappa(\omega)$	ω	$\kappa(\omega)$	ω	$\kappa(\omega)$
$pbfwk$	2	$p\overline{b}fwk$	5	$\overline{p}bfwk$	0	$\overline{p}\overline{b}fwk$	1
$pbfw\overline{k}$	2	$p\overline{b}fw\overline{k}$	4	$\overline{p}bfw\overline{k}$	0	$\overline{p}\overline{b}fw\overline{k}$	0
$pbf\overline{w}k$	3	$p\overline{b}f\overline{w}k$	5	$\overline{p}bf\overline{w}k$	1	$\overline{p}\overline{b}f\overline{w}k$	1
$pbf\overline{w}\overline{k}$	3	$p\overline{b}f\overline{w}\overline{k}$	4	$\overline{p}bf\overline{w}\overline{k}$	1	$\overline{p}\overline{b}f\overline{w}\overline{k}$	0
$pb\overline{f}wk$	1	$p\overline{b}\overline{f}wk$	3	$\overline{p}b\overline{f}wk$	1	$\overline{p}\overline{b}\overline{f}wk$	1
$pb\overline{f}w\overline{k}$	1	$p\overline{b}\overline{f}w\overline{k}$	2	$\overline{p}b\overline{f}w\overline{k}$	1	$\overline{p}\overline{b}\overline{f}w\overline{k}$	0
$pb\overline{f}\overline{w}k$	2	$p\overline{b}\overline{f}\overline{w}k$	3	$\overline{p}b\overline{f}\overline{w}k$	2	$\overline{p}\overline{b}\overline{f}\overline{w}k$	1
$pb\overline{f}\overline{w}\overline{k}$	2	$p\overline{b}\overline{f}\overline{w}\overline{k}$	2	$\overline{p}b\overline{f}\overline{w}\overline{k}$	2	$\overline{p}\overline{b}\overline{f}\overline{w}\overline{k}$	0

Fig. 1. Ranking function κ accepting the rule set $\mathcal{R}$ given in Example 1.

4 Specification of Ranking Functions as Solutions of a Constraint Satisfaction Problem

Given a set $\mathcal{R} = \{R_1, \ldots, R_n\}$ of conditionals, a ranking function κ that accepts every R_i represents an epistemic state of an agent accepting $\mathcal{R}$. If there is no κ that accepts every R_i then $\mathcal{R}$ is *inconsistent*. For the rest of this paper, we assume that $\mathcal{R}$ is consistent.

For any consistent $\mathcal{R}$ there may be many different κ accepting $\mathcal{R}$, each representing a complete set of beliefs with respect to every possible formula A and every conditional $(B|A)$. Thus, every such κ inductively completes the knowledge given by $\mathcal{R}$, and it is a vital question whether some κ' is to be preferred to some other κ'', or whether there is a unique "best" κ. Different ways of determining a ranking function are given by *system Z* [9,10] or its more sophisticated extension *system Z** [9], see also [6]; for an approach using rational world rankings see [19]. For quantitative knowledge bases of the form $\mathcal{R}_x = \{(B_1|A_1)[x_1], \ldots, (B_n|A_n)[x_n]\}$ with probability values x_i and with models being probability distributions P satisfying a probabilistic conditional $(B_i|A_i)[x_i]$ iff $P(B_i|A_i) = x_i$, a unique model can be choosen by employing the principle of maximum entropy [11,16,17]; the maximum entropy model is a best model in the sense that it is the most unbiased one among all models satisfying $\mathcal{R}_x$.

Using the maximum entropy idea, in [13] a generalization of system Z* is suggested. Based on an algebraic treatment of conditionals, the notion of *conditional indifference* of κ with respect to $\mathcal{R}$ is defined and the following criterion for conditional indifference is given: An OCF κ is indifferent with respect to $\mathcal{R} = \{(B_1|A_1), \ldots, (B_n|A_n)\}$ iff $\kappa(A_i) < \infty$ for all $i \in \{1, \ldots, n\}$ and there are rational numbers $\kappa_0, \kappa_i^+, \kappa_i^- \in \mathbb{Q}$, $1 \leqslant i \leqslant n$, such that for all $\omega \in \Omega$,

$$\kappa(\omega) = \kappa_0 + \sum_{\substack{1 \leqslant i \leqslant n \\ \omega \models A_i B_i}} \kappa_i^+ + \sum_{\substack{1 \leqslant i \leqslant n \\ \omega \models A_i \overline{B_i}}} \kappa_i^- . \tag{5}$$

When starting with an epistemic state of complete ignorance (i.e., each world ω has rank 0), for each rule $(B_i|A_i)$ the values κ_i^+, κ_i^- determine how the rank of each satisfying world and of each falsifying world, respectively, should be changed:

- If the world ω verifies the conditional $(B_i|A_i)$, – i.e., $\omega \models A_iB_i$ –, then κ_i^+ is used in the summation to obtain the value $\kappa(\omega)$.
- Likewise, if ω falsifies the conditional $(B_i|A_i)$, – i.e., $\omega \models A_i\overline{B_i}$ –, then κ_i^- is used in the summation instead.
- If the conditional $(B_i|A_i)$ is not applicable in ω, – i.e., $\omega \models \overline{A_i}$ –, then this conditional does not influence the value $\kappa(\omega)$.

κ_0 is a normalization constant ensuring that there is a smallest world rank 0. Employing the postulate that the ranks of a satisfying world should not be changed and requiring that changing the rank of a falsifying world may not result in an increase of the world's plausibility leads to the concept of a *c-representation* [12,13]:

Definition 1. *Let $\mathcal{R} = \{(B_1|A_1), \ldots, (B_n|A_n)\}$. Any ranking function κ satisfying the conditional indifference condition (5) and $\kappa_i^+ = 0$, $\kappa_i^- \geqslant 0$ (and thus also $\kappa_0 = 0$ since $\mathcal{R}$ is assumed to be consistent) as well as*

$$\kappa(A_iB_i) < \kappa(A_i\overline{B_i}) \tag{6}$$

for all $i \in \{1, \ldots, n\}$ is called a (special) c-representation *of $\mathcal{R}$.*

Note that for $i \in \{1, \ldots, n\}$, condition (6) expresses that κ accepts the conditional $R_i = (B_i|A_i) \in \mathcal{R}$ (cf. the definition of the satisfaction relation in (4)) and that this also implies $\kappa(A_i) < \infty$.

Thus, finding a c-representation for $\mathcal{R}$ amounts to choosing appropriate values $\kappa_1^-, \ldots, \kappa_n^-$. In [2] this situation is formulated as a constraint satisfaction problem $CR(\mathcal{R})$ whose solutions are vectors of the form $(\kappa_1^-, \ldots, \kappa_n^-)$ determining c-representations of $\mathcal{R}$. The development of $CR(\mathcal{R})$ exploits (2) and (5) to reformulate (6) and requires that the κ_i^- are natural numbers (and not just rational numbers). In the following, we set $\min(\emptyset) = \infty$.

Definition 2. *[$CR(\mathcal{R})$] Let $\mathcal{R} = \{(B_1|A_1), \ldots, (B_n|A_n)\}$. The constraint satisfaction problem for c-representations of $\mathcal{R}$, denoted by $CR(\mathcal{R})$, is given by the conjunction of the constraints*

$$\kappa_i^- \geqslant 0 \tag{7}$$

$$\kappa_i^- > \min_{\omega \models A_iB_i} \sum_{\substack{j \neq i \\ \omega \models A_j\overline{B_j}}} \kappa_j^- - \min_{\omega \models A_i\overline{B_i}} \sum_{\substack{j \neq i \\ \omega \models A_j\overline{B_j}}} \kappa_j^- \tag{8}$$

for all $i \in \{1, \ldots, n\}$.

A solution of $CR(\mathcal{R})$ is an n-tuple $(\kappa_1^-, \ldots, \kappa_n^-)$ of natural numbers, and with $Sol_{CR}(\mathcal{R})$ we denote the set of all solutions of $CR(\mathcal{R})$.

Proposition 1. *For $\mathcal{R} = \{(B_1|A_1), \ldots, (B_n|A_n)\}$ let $\vec{\kappa} = (\kappa_1^-, \ldots, \kappa_n^-) \in Sol_{CR}(\mathcal{R})$. Then the function κ defined by*

$$\kappa(\omega) = \sum_{\substack{1 \leqslant i \leqslant n \\ \omega \models A_i \overline{B_i}}} \kappa_i^- \tag{9}$$

in the following denoted by $\kappa_{\vec{\kappa}}$, is an OCF that accepts $\mathcal{R}$.

All c-representations built from (7), (8), and (9) provide an excellent basis for model-based inference [12,13]. However, from the point of view of minimal specificity (see e.g. [4]), those c-representations with minimal κ_i^- yielding minimal degrees of implausibility are most interesting.

While different orderings on $Sol_{CR}(\mathcal{R})$ can be defined, leading to different minimality notions, in the following we will first focus on the ordering on $Sol_{CR}(\mathcal{R})$ induced by taking the sum of the κ_i^-, i.e.

$$(\kappa_1^-, \ldots, \kappa_n^-) \preccurlyeq_+ ({\kappa'}_1^-, \ldots, {\kappa'}_n^-) \quad \text{iff} \quad \sum_{1 \leqslant i \leqslant n} \kappa_i^- \leqslant \sum_{1 \leqslant i \leqslant n} {\kappa'}_i^- . \tag{10}$$

A vector $\vec{\kappa}$ is *sum-minimal* (just called *minimal* in the following) iff there is no vector $\vec{\kappa}'$ such that $\vec{\kappa}' \prec_+ \vec{\kappa}$ where $\prec_+$ is the irreflexive subrelation of $\preccurlyeq_+$. As we are interested in minimal κ_i^--vectors, an important question is whether there is always a unique minimal solution. This is not the case; the following example that is also discussed in [15] illustrates that $Sol_{CR}(\mathcal{R})$ may have more than one minimal element.

Example 2. Let $\mathcal{R}_{birds} = \{R_1, R_2, R_3\}$ be the following set of conditionals:

$R_1 : (f\|b)$	*$\underline{b}$irds $\underline{f}$ly*
$R_2 : (a\|b)$	*$\underline{b}$irds are $\underline{a}$nimals*
$R_3 : (a\|fb)$	*$\underline{f}$lying $\underline{b}$irds are $\underline{a}$nimals*

From (8) we get

$$\begin{aligned} \kappa_1^- &> 0 \\ \kappa_2^- &> 0 - min\{\kappa_1^-, \kappa_3^-\} \\ \kappa_3^- &> 0 - \kappa_2^- \end{aligned}$$

and since $\kappa_i^- \geqslant 0$ according to (7), the two vectors

$$\begin{aligned} sol_1 &= (\kappa_1^-, \kappa_2^-, \kappa_3^-) = (1, 1, 0) \\ sol_2 &= (\kappa_1^-, \kappa_2^-, \kappa_3^-) = (1, 0, 1) \end{aligned}$$

are two different solutions of $CR(\mathcal{R}_{birds})$ with $\sum_{1 \leqslant i \leqslant n} \kappa_i^- = 2$ that are both minimal in $Sol_{CR}(\mathcal{R}_{birds})$ with respect to $\preccurlyeq_+$.

5 A Declarative CLP Program for $CR(\mathcal{R})$

In this section, we will develop a CLP program `GenOCF` solving $CR(\mathcal{R})$. Our main objective is to obtain a declarative program that is as close as possible to the abstract formulation of $CR(\mathcal{R})$ while exploiting the concepts of constraint logic programming. We will employ finite domain constraints, and from (7) we immediately get a lower bound for κ_i^-. Considering that we are interested mainly in minimal solutions, due to (8) we can safely restrict ourselves to n as an upper bound for κ_i^-, yielding

$$0 \leqslant \kappa_i^- \leqslant n \tag{11}$$

for all $i \in \{1, \ldots, n\}$ with n being the number of conditionals in $\mathcal{R}$.

5.1 Input Format and Preliminaries

Since we want to focus on the constraint solving part, we do not consider reading and parsing a knowledge base $\mathcal{R} = \{(B_1|A_1), \ldots, (B_n|A_n)\}$. Instead, we assume that $\mathcal{R}$ is already given as a Prolog code file providing the following predicates `variables/1`, `conditional/3` and `indices/1`:

`variables`([$\mathtt{a}_1, \ldots, \mathtt{a}_m$])	% list of atoms in Σ
`conditional`($i, \langle A_i\rangle, \langle B_i\rangle$)	% representation of ith conditional $(B_i\|A_i)$
`indices`([1,...,n])	% list of indices $\{1, \ldots, n\}$

If $\Sigma = \{a_1, \ldots, a_m\}$ is the set of atoms, we assume a fixed ordering $a_1 < a_2 < \ldots < a_m$ on Σ given by the predicate `variables`([a_1,...,a_m]). The fixed index ordering gven by `indices`([1,...,n]) ensures that the conditionals are ordered consecutively from 1 to n. Thus, the i-th conditional can be accessed by `conditional`(i, `A,B`), and in a solution vector [$\mathtt{K}_1$,...,$\mathtt{K}_n$], the i-th component $\mathtt{K_i}$ is the κ^--value for the i-th conditional.

In the representation of a conditional, a propositional formula A, constituting the antecedent or the consequence of the conditional, is represented by $\langle A\rangle$ where $\langle A\rangle$ is a Prolog list [$\langle D_1\rangle$,...,$\langle D_l\rangle$]. Each $\langle D_i\rangle$ represents a conjunction of literals such that $D_1 \vee \ldots \vee D_l$ is a disjunctive normal form of A.

Each $\langle D\rangle$, representing a conjunction of literals, is a Prolog list [b_1,...,b_m] of fixed length m where m is the number of atoms in Σ and with $b_k \in \{\mathtt{0}, \mathtt{1}, \_\}$. Such a list [$b_1$,...,$b_m$] represents the conjunctions of atoms obtained from $\dot{a}_1 \wedge \dot{a}_2 \wedge \ldots \wedge \dot{a}_m$ by eliminating all occurrences of $\top$, where

$$\dot{a}_k = \begin{cases} a_k & \text{if } \ b_k = 1 \\ \overline{a_k} & \text{if } \ b_k = 0 \\ \top & \text{if } \ b_k = \_ \end{cases}$$

Example 3. The internal representation of the knowledge base presented in Example 1. is shown in Fig. 2.

```
variables([p,b,f,w,k]).

%                  p b f w k      p b f w k
conditional(1,[[_,1,_,_,_]],[[_,_,1,_,_]]).  % (f|b)  birds fly
conditional(2,[[1,_,_,_,_]],[[_,1,_,_,_]]).  % (b|p)  penguins are birds
conditional(3,[[1,_,_,_,_]],[[_,_,0,_,_]]).  % (-f|p) penguins do not fly
conditional(4,[[_,1,_,_,_]],[[_,_,_,1,_]]).  % (w|b)  birds have wings
conditional(5,[[_,_,_,_,1]],[[_,1,_,_,_]]).  % (b|k)  kiwis are birds

indices([1,2,3,4,5]).
```

Fig. 2. Internal representation of the knowledge base from Example 1.

As further preliminaries, using `conditional/3` and `indices/1`, we have implemented the predicates `worlds/1`, `verifying_worlds/2`, `falsifying_worlds/2`, and `falsify/2`, realising the evaluation of the indicator function (1) given in Sect. 2:

`worlds`(Ws)	% *Ws* list of possible worlds
`verifying_worlds`(i, Ws)	% *Ws* list of worlds verifying ith conditional
`falsifying_worlds`(i, Ws)	% *Ws* list of worlds falsifying ith conditional
`falsify`(i, W)	% world W falsifies ith conditional

where worlds are represented as complete conjunctions of literals over Σ, using the representation described above.

Using these predicates, in the following subsections we will present the complete source code of the constraint logic program `GenOCF` solving $CR(\mathcal{R})$.

5.2 Generation of Constraints

The particular program code given here uses the SICStus Prolog system[1] and its clp(fd) library implementing constraint logic programming over finite domains [14].

The main predicate `kappa/2` expecting a knowledge base `KB` of conditionals and yielding a vector `K` of κ_i^- values as specified by (8) is presented in Fig. 3.

After reading in the knowledge base, the constraints for `K` are generated. In `constraints/1`, after getting the list of indices, a list `K` of free constraint variables, one for each conditional, is generated; in the two subsequent subgoals, the constraints corresponding to the formulas (11) and (8) are generated, constraining the elements of `K` accordingly. Finally, `labeling([], K)` yields a list of κ_i^- values. Upon backtracking, this will enumerate all possible solutions with an upper bound of n as in (11) for each κ_i^-. Later on, we will demonstrate how to modify `kappa/2` in order to take minimality into account (Sect. 5.3).

How the subgoal `constrain_K(Is, K)` in `kappa/2` generates a constraint for each index $i \in \{1, \dots, n\}$ according to (8) is defined in Fig. 4.

[1] http://www.sics.se/isl/sicstuswww/site/index.html

```
kappa(KB, K) :-           % K is kappa vector of c-representation for KB
   consult(KB),
   constraints(K),        % generate constraints for K
   labeling([], K).       % generate solution

constraints(K) :-
   indices(Is),           % get list of indices [1,2,...,N]
   length(Is, N),         % N number of conditionals in KB
   length(K, N),          % generate K = [Kappa_1,...,Kappa_N] of free var.
   domain(K, 0, N),       % 0 <= kappa_I <= N  for all I according to (11)
   constrain_K(Is, K).    % generate constraints according to (8)
```

Fig. 3. Main predicate `kappa/2`

```
constrain_K([],_).                         % generate constraints for
constrain_K([I|Is],K) :-                   % all kappa_I as in (8)
   constrain_Ki(I,K), constrain_K(Is,K).

constrain_Ki(I,K) :-               % generate constraint for kappa_I as in (8)
   verifying_worlds(I, VWorlds),      % all worlds verifying I-th condit.
   falsifying_worlds(I, FWorlds),     % all worlds falsifying I-th condit.
   list_of_sums(I, K, VWorlds, VS),   % VS list of sums for verif. worlds
   list_of_sums(I, K, FWorlds, FS),   % FS list of sums for falsif. worlds
   minimum(Vmin, VS),                 % Vmin minium for verifying worlds
   minimum(Fmin, FS),                 % Fmin minium for falsifying worlds
   element(I, K, Ki),                 % Ki constraint variable for kappa_I
   Ki #> Vmin - Fmin.                 % constraint for kappa_I as in (8)
```

Fig. 4. Constraining the vector `K` representing $\kappa_1^-, \ldots, \kappa_n^-$ as in (8)

Given an index `I`, `constrain_Ki(I,K)` (cf. Fig. 4) determines all worlds verifying and falsifying the `I`-th conditional; over these two sets of worlds the two min expressions in (8) are defined. Two lists `VS` and `FS` of sums corresponding exactly to the first and the second sum, repectively, in (8) are generated (how this is done is defined in Fig. 5 and will be explained below). With the constraint variables `Vmin` and `Fmin` denoting the minimum of these two lists, the constraint

$$\texttt{Ki \#> Vmin} - \texttt{Fmin}$$

given in the last line of Fig. 4 reflects precisely the restriction on κ_i^- given by (8).

For an index `I`, a kappa vector `K`, and a list of worlds `Ws`, the goal `list_of_sums(I, K, Ws, Ss)` (cf. Fig. 5) yields a list `Ss` of sums such that for each world `W` in `Ws`, there is a sum `S` in `Ss` that is generated by `sum_kappa_j(Js, I, K, W, S)` where `Js` is the list of indices $\{1, \ldots, n\}$. In the goal `sum_kappa_j (Js, I, K, W, S)`, `S` corresponds exactly to the respective sum expression in (8), i.e., it is the sum of all `Kj` such that `J` $\neq$ `I` and `W` falsifies the `j`-th conditional.

```
% list_of_sums(I, K, Ws, Ss)  generates list of sums as in (8):
%     I   index from 1,...,N
%     K   kappa vector
%     Ws  list of worlds
%     Ss  list of sums:
%         for each world W in Ws there is S in Ss s.t.
%            S is sum of all kappa_J with
%            J \= I and W falsifies J-th conditional
list_of_sums(_, _, [], []).
list_of_sums(I, K, [W|Ws], [S|Ss]) :-
   indices(Js),
   sum_kappa_j(Js, I, K, W, S),
   list_of_sums(I, K, Ws, Ss).

% sum_kappa_j(Js, I, K, W, S)  generates a sum as in (8):
%     Js  list of indices [1,...,N]
%     I   index from 1,...,N
%     K   kappa vector
%     W   world
%     S   sum of all kappa_J s.t.
%            J \= I and W falsifies J-th conditional
sum_kappa_j([], _, _, _, 0).
sum_kappa_j([J|Js], I, K, W, S) :-
    sum_kappa_j(Js, I, K, W, S1),
    element(J, K, Kj),
    ((J \= I, falsify(J, W)) -> S #= S1 + Kj; S #= S1).
```

Fig. 5. Generating list of sums of κ_i^- as in (8)

Example 4. Suppose that `kb_birds.pl` is a file containing the conditionals of the knowledge base $\mathcal{R}_{birds}$ given in Example 2.. Then the first five solutions generated by the program given in Figs. 3, 4, 5 are:

```
| ?- kappa('kb_birds.pl', K).
K = [0,1,1] ? ;
K = [1,0,2] ? ;
K = [1,0,3] ? ;
K = [1,1,0] ? ;
K = [1,1,1] ?
```

Note that the first and the fourth solution are the minimal solutions.

Example 5. If `kb_penguins.pl` is a file containing the conditionals of the knowledge base $\mathcal{R}$ given in Example 1., the first six solutions generated by `kappa/2` are:

```
| ?- kappa('kb_penguins.pl', K).
K = [1,2,2,1,1] ? ;
K = [1,2,2,1,2] ? ;
K = [1,2,2,1,3] ? ;
```

```
kappa_min_sum(KB, K) :-         % K is minimal vector for KB, one solution
   consult(KB),
   constraints(K),              % generate constraints for K
   sum(K, #=, S),               % constraint variable S for sum of kappa_I
   minimize(labeling([],K), S). % generate single minimal solution
```

Fig. 6. Predicate `kappa_min_sum/2` generating a minimal solution

```
kappa_min_sum_all(KB, K) :-    % K is minimal vector for KB, all solutions
   consult(KB),
   constraints(K),             % generate constraints for K
   sum(K, #=, S),              % constraint variable S for sum of kappa_I
   min_sum_kappas(K, S),       % determine minimal value for S
   labeling([], K).            % generate all minimal solutions

min_sum_kappas(K, Min) :-      % Min is sum of a minimal solution for K
   once((labeling([up],[Min]),
         \+ \+ labeling([],K))).
```

Fig. 7. Predicate `kappa_min_all/2` generating exactly all minimal solutions

```
K = [1,2,2,1,4] ? ;
K = [1,2,2,1,5] ? ;
K = [1,2,2,2,1] ?
```

5.3 Generation of Minimal Solutions

The enumeration predicate `labeling/2` of SICStus Prolog allows for an option that minimizes the value of a cost variable. Since we are aiming at minimizing the sum of all κ_i^-, the constraint `sum(K, #=, S)` introduces such a cost variable `S`. Thus, exploiting the SICStus Prolog minimization feature, we can easily modify `kappa/2` to generate a minimal solution: We just have to replace the last subgoal `labeling([], K)` in Fig. 3 by the two subgoals given in Fig. 6.

With this modification, the obtained predicate `kappa_min_sum/2` returns a single minimal solution (and fails on backtracking). Hence calling `?- kappa_min_sum('kb_birds.pl', K).` similar as in Example 4. yields the minimal solution `K = [0,1,1]`.

However, as pointed out in Sect. 4, there are good reasons for considering not just a single minimal solution, but all minimal solutions. We can achieve the computation of all minimal solutions by another slight modification of `kappa/2`. This time, the enumeration subgoal `labeling([], K)` in Fig. 3 is preceded by two new subgoals as in `kappa_min_sum_all/2` in Fig. 7.

The first new subgoal `sum(K, #=, S)` introduces a constraint variable `S` just as in `kappa_min_sum/2`. In the subgoal `min_sum_kappas(K, S)`, this variable `S` is constrained to the sum of a minimal solution as determined by `min_sum_kappas(K, Min)`. These two new subgoals ensure that in the generation caused by the final subgoal `labeling([], K)`, exactly all minimal solutions are enumerated.

```
rank(W, K, R) :-                    % the rank of world W under K is R
   findall(C, falsify(C,W), Cs),
   sum_kappa(Cs, K, R).             % determine R according to (9)

sum_kappa([], _, 0).                % sum of kappa-i-minus values for
sum_kappa([C|Cs], K, R) :-          % falsifying conditionals according
   sum_kappa(Cs, K, R1),            % to equation (9)
   element(C, K, Kj),
   R #= R1 + Kj.
```

Fig. 8. Predicate `rank/3` determining the rank of a world for given

Example 6. Continuing Example 4., calling

```
| ?- kappa_min_sum_all('kb_birds.pl', K).
K = [0,1,1] ? ;
K = [1,1,0] ? ;
no
```

yields the two minimal solutions for $\mathcal{R}_{birds}$.

Example 7. For the situation in Example 5., `kappa_min_sum_all/2` reveals that there is a unique minimal solution:

```
| ?- kappa_min_sum_all('kb_penguins.pl', K).
K = [1,2,2,1,1] ? ;
no
```

The predicate `rank/3` given in Fig. 8 can be used for determining the OCF $\kappa_{\vec{\kappa}}$ induced by the vector $\vec{\kappa} = (\kappa_1^-, \kappa_2^-, \kappa_3^-, \kappa_4^-, \kappa_5^-) = (1, 2, 2, 1, 1)$ according to (9), yielding the ranking function given in Fig. 1.

5.4 Alternative Notions of Minimality

In general, it is still an open problem how to strengthen the requirements defining a c-representation so that a unique minimal solution is guaranteed to exist. Such a strengthening could use alternative minimality criteria. In this subsection, we illustrate the realization of an alternative minimality criterion in our constraint logic programming approach.

Instead of ordering the vectors $\vec{\kappa}$ by the sum of their components as done by $\preccurlyeq_+$ in (10), we could consider a componentwise ordering $\preccurlyeq_{cw}$

$$(\kappa_1^-, \dots, \kappa_n^-) \preccurlyeq_{cw} ({\kappa'}_1^-, \dots, {\kappa'}_n^-) \quad \text{iff} \quad \kappa_i^- \leqslant {\kappa'}_i^- \text{ for all } \; i \in \{1, \dots, n\} \tag{12}$$

yielding a partial order $\preccurlyeq_{cw}$ on $Sol_{CR}(\mathcal{R})$.

Let $\prec_{cw}$ denote the irreflexive subrelations of $\preccurlyeq_{cw}$, respectively. A vector $\vec{\kappa}$ is *componentwise minimal* (or *cw-minimal*) iff there is no vector $\vec{\kappa}'$ with $\vec{\kappa}' \prec_{cw} \vec{\kappa}$. In order to demonstrate the flexibility of the high-level CLP implementation

```
kappa_min_cw_all(KB, K) :-          % all cw-minimal solutions K for KB
   kappa(KB, K),                    % K is kappa vector for KB
   minimal_cw(K).                   % K is minimal componentwise

minimal_cw(K) :-                    % K is cw-minimal
   constraints(K2),                 % if there is no kappa vector K2 that
   \+ once((lt_cw(K2, K),           % is componentwise strictly less than K
            labeling([],K2))).

lt_cw([K1|K1s], [K2|K2s]) :-        % one vector component in first
   (K1 #< K2, leq_cw(K1s, K2s));    % argument is strictly less than the
   (K1 #= K2, lt_cw(K1s, K2s)).     % corresponding component in second
                                    % argument, all other are less or equal

leq_cw([], []).                     % every vector component in
leq_cw([K1|K1s], [K2|K2s]) :-       % first argument less or equal to
   K1 #=< K2,                       % corresponding component in
   leq_cw(K1s, K2s).                % second argument
```

Fig. 9. Predicate `kappa_min_cw_all` computing all componentwise minimal solutions

realized in `GenOCF`, we will show how slight modifications of the program realize these alternative notions of minimality.

The predicate `kappa_min_cw_all/2` as given in Fig. 9 computes all componentwise minimal solution for a knowledge base. After consulting the knowledge base `KB`, the subgoal `kappa(KB, K)` says that `K` is a solution for `KB`, while `minimal_cw(K)` ensures that `K` is cw-minimal. The predicate `minimal_cw/1` enforces that there is no solution vector `K2` for the given knowledge base that is strictly less than `K`: If `constraints(K2)` succeeds where `constraints/1` is given as in Fig. 3, then there is no labeling for `K2` under the constraint `lt_cw(K2,K)`. The predicate `lt_cw/2` takes two vectors of the same length and succeeds if there is at least one position where the component at that position in the first vector is strictly less than the corresponding component in the second vector (ensured by the constraint with `#<` in `lt_cw/2` in Fig. 9), and all other corresponding vector components are less or equal (ensured by the constraints with `#=` in `lt_cw/2` and with `#=<` in `leq_cw/2`).

Example 8. Continuing Example 6., calling

```
| ?- kappa_min_cw_all('kb_birds.pl', K).
K = [1,0,1] ? ;
K = [1,1,0] ? ;
no
```

reveals that in this simple example the set of sum-minimal solutions coincides with the set of cw-minimal solutions.

In our further invstigations, we will extend `GenOCF` to be able to take into account more alternative minimality criteria. For instance, as illustrated in the previous example, GenOCF determines that both $\kappa_1 = [\mathtt{1}, \mathtt{0}, \mathtt{1}]$ and $\kappa_2 = [\mathtt{1}, \mathtt{1}, \mathtt{0}]$ are minimal with respect to $\preccurlyeq_+$ and also with respect to $\preccurlyeq_{cw}$ for $\mathcal{R}_{birds}$. However, we could also compare the full OCFs induced by κ_1 and κ_2 according to (9). These induced OCFs are given by the following table:

ω	$\kappa_1(\omega)$	$\kappa_2(\omega)$	ω	$\kappa_1(\omega)$	$\kappa_2(\omega)$
fba	0	0	$\overline{f}ba$	1	1
$fb\overline{a}$	1	1	$\overline{f}b\overline{a}$	1	2
$f\overline{b}a$	0	0	$\overline{f}\,\overline{b}a$	0	0
$f\overline{b}\overline{a}$	0	0	$\overline{f}\,\overline{b}\overline{a}$	0	0

Note that under κ_1, the world $\overline{f}b\overline{a}$ has a smaller rank than under κ_2, while all other worlds have the same rank under both OCFs. Further theoretical and experimental studies are needed for this and still other minimality criteria.

6 Example Applications and First Evaluation

Although the objective in developing GenOCF was on being as close as possible to the abstract formulation of the constraint satisfaction problem $CR(\mathcal{R})$, we will present the results of some first example applications we have carried out.

For $n \geqslant 1$, we generated synthetic knowledge bases kb_synth<n>_c<2n $-$ 1>.pl according to the following schema: Using the variables $\{f\} \cup \{a_1, \ldots, a_n\}$, kb_synth<$n$>_c<2n $-$ 1>.pl contains the $2 * n - 1$ conditionals given by::

$$\begin{array}{lll} (f|a_i) & \text{if } i & \text{is odd, } i \in \{1, \ldots, n\} \\ (\overline{f}|a_i) & \text{if } i & \text{is even, } i \in \{1, \ldots, n\} \\ (a_i|a_{i+1}) & \text{if } i & \in \{1, \ldots, n-1\} \end{array}$$

For instance, kb_synth4_c7.pl uses the five variables $\{f, a_1, a_2, a_3, a_4\}$ and contains the seven conditionals:

$$\begin{array}{c} (f|a_1) \\ (\overline{f}|a_2) \\ (f|a_3) \\ (\overline{f}|a_4) \\ (a_1|a_2) \\ (a_2|a_3) \\ (a_3|a_4) \end{array}$$

The basic idea underlying the construction of these synthetic knowledge bases kb_synth<n>_c<2n$-$1>.pl is to establish a kind of subclass relationship between a_{i+1} and a_i for each $i \in \{1, \ldots, n-1\}$ on the one hand, and to state that every a_{i+1} is exceptional to a_i with respect to its behaviour regarding f, again for each $i \in \{1, \ldots, n-1\}$. This sequence of pairwise exceptional elements will force any minimal solution of $CR($kb_synth<n>_c<2n$-$1>.pl$)$ to have at least one κ_i^- value of size greater or equal to n.

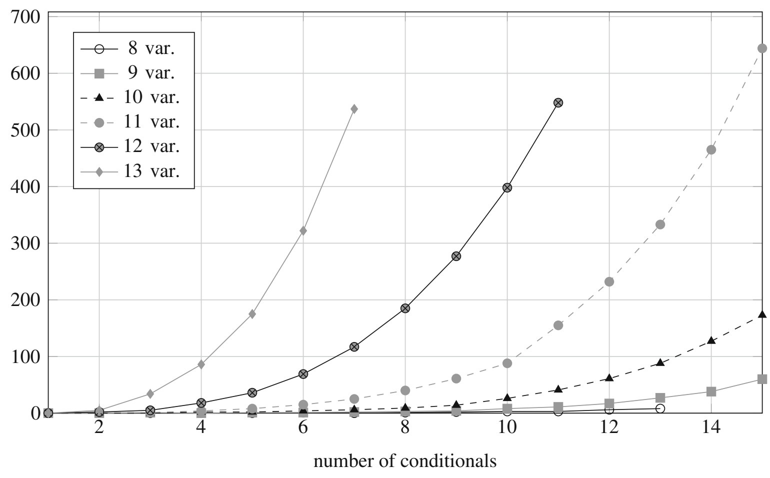

Fig. 10. Execution times (given in seconds) of `GenOCF` under SICStus Prolog for computing all sum-minimal solutions for various synthesized knowledge bases

From `kb_synth<`n`>_c<`m`>.pl`, the knowledge bases `kb_synth<`n`>_c<`$m–j$`>.pl` are generated for $j \in \{1, \dots, m-1\}$ by removing the last j conditionals. For instance, `kb_synth4_c5.pl` is obtained from `kb_synth4_c7.pl` by removing the two conditionals $\{(a_2|a_3),\ (a_3|a_4)\}$.

Figure 10 shows the time needed by `GenOCF` for computing all sum-minimal solutions for some of these synthesized knowledge bases with different numbers of variables and conditionals. Execution times are given in seconds for measurements taken in the following environment: SICStus 4.0.8 (x86-linux-glibc2.3), Intel Core 2 Duo E6850 3.00GHz.

Of course, these knowledge bases are by no means representative, and further evaluation is needed. In particular, investigating the complexity depending on the number of variables and conditionals and determining an upper bound for worst-case complexity has still to be done; the graphical illustration in Fig. 10 clearly indicates an exponential increase. However, it should be noted that the high-level, declarative approach taken here provides many opportunities for improving run-time efficiency. For instance, it suffices to compute the verifying and the falsifying worlds for each conditional only once instead of multiple times when generating the constraints for a solution vector `K` as done in Fig. 4. Furthermore, while the code for `GenOCF` given above uses SICStus Prolog, we also have a variant of `GenOCF` for the SWI Prolog system[2] [20]. In our further investigations, we want to evaluate `GenOCF` also using SWI Prolog, to elaborate the changes required and the options provided when moving between SICStus and SWI Prolog, and to study whether there are any significant differences in execution that might depend on the two different Prolog systems and their options.

[2] http://www.swi-prolog.org/index.html

7 Conclusions and Further Work

While for a set of probabilistic conditionals $(B_i|A_i)[x_i]$ the principle of maximum entropy yields a unique model, for a set $\mathcal{R}$ of qualitative default rules $(B_i|A_i)$ there may be several minimal ranking functions. In this paper, we developed a CLP approach for solving $CR(\mathcal{R})$, realized in the Prolog program `GenOCF`. The solutions of the constraint satisfaction problem $CR(\mathcal{R})$ are vectors of natural numbers $\vec{\kappa} = (\kappa_1^-, \ldots, \kappa_n^-)$ that uniquely determine an OCF $\kappa_{\vec{\kappa}}$ accepting all conditionals in $\mathcal{R}$. `GenOCF` is also able to generate exactly all minimal solutions of $CR(\mathcal{R})$ for different notions of minimality. Minimal solutions of $CR(\mathcal{R})$ are of special interest for model-based inference.

In general, it is an open problem how to strengthen the requirements defining a c-representation so that a unique solution is guaranteed to exist. The declarative nature of constraint logic programming supports easy constraint modification, enabling the experimentation and practical evaluation of different notions of minimality for $Sol_{CR}(\mathcal{R})$ and of additional requirements that might be imposed on a ranking function. Furthermore, in [8] the framework of default rules concidered here is extended by allowing not only default rules in the knowledge base $\mathcal{R}$, but also strict knowledge, rendering some worlds completely impossibe. This can yield a reduction of the problem's complexity, and it will be interesting to see which effects the incorporation of strict knowledge will have on the CLP approach presented here.

References

1. Beierle, C., Kern-Isberner, G.: A verified AsmL implementation of belief revision. In: Börger, E., Butler, M., Bowen, J.P., Boca, P. (eds.) ABZ 2008. LNCS, vol. 5238, pp. 98–111. Springer, Heidelberg (2008)
2. Beierle, C., Kern-Isberner, G.: On the computation of ranking functions for default rules - a challenge for constraint programming. In: Heiß, H.-U., Pepper, P., Schlingloff, H., Schneider, J. (eds.) Informatik 2011: Informatik schafft Communities, Beiträge der 41. Jahrestagung der Gesellschaft für Informatik e.V. (GI), 4.-7.10.2011, Berlin (Abstract Proceedings), volume P-192 of LNI. GI (2011)
3. Beierle, C., Kern-Isberner, G., Koch, N.: A high-level implementation of a system for automated reasoning with default rules (system description). In: Armando, A., Baumgartner, P., Dowek, G. (eds.) IJCAR 2008. LNCS (LNAI), vol. 5195, pp. 147–153. Springer, Heidelberg (2008)
4. Benferhat, S., Dubois, D., Prade, H.: Representing default rules in possibilistic logic. In: Proceedings 3th International Conference on Principles of Knowledge Representation and Reasoning KR'92, pp. 673–684 (1992)
5. Bourne, R.A.: Default reasoning using maximum entropy and variable strength defaults. Ph.D. thesis, Univ. of London (1999)
6. Bourne, R.A., Parsons, S.: Maximum entropy and variable strength defaults. In: Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, IJCAI'99, pp. 50–55 (1999)
7. DeFinetti, B.: Theory of Probability, vol. 1,2. Wiley, New York (1974)
8. Eiter, T., Lukasiewicz, T.: Complexity results for structure-based causality. Artif. Intell. **142**(1), 53–89 (2002)

9. Goldszmidt, M., Morris, P., Pearl, J.: A maximum entropy approach to nonmonotonic reasoning. IEEE Trans. Pattern Anal. Mach. Intell. **15**(3), 220–232 (1993)
10. Goldszmidt, M., Pearl, J.: Qualitative probabilities for default reasoning, belief revision, and causal modeling. Artif. Intell. **84**, 57–112 (1996)
11. Kern-Isberner, G.: Characterizing the principle of minimum cross-entropy within a conditional-logical framework. Artif. Intell. **98**, 169–208 (1998)
12. Kern-Isberner, G.: Conditionals in nonmonotonic reasoning and belief revision. LNCS (LNAI), vol. 2087. Springer, Heidelberg (2001)
13. Kern-Isberner, G.: Handling conditionals adequately in uncertain reasoning and belief revision. J. Appl. Non-Class. Logics **12**(2), 215–237 (2002)
14. Carlsson, M., Ottosson, G., Carlson, B.: An open-ended finite domain constraint solver. In: Glaser, H., Hartel, P.H., Kuchen, H. (eds.) PLILP 1997. LNCS, vol. 1292, pp. 191–206. Springer, Heidelberg (1997)
15. Müller, C.: Implementing default rules by optimal conditional ranking functions. B.Sc. Thesis, Department of Computer Science, FernUniversität in Hagen, Germany (2004) (in German)
16. Paris, J.B.: The uncertain reasoner's companion - A mathematical perspective. Cambridge University Press, Cambridge (1994)
17. Paris, J.B., Vencovska, A.: In defence of the maximum entropy inference process. Int. J. Approximate Reasoning **17**(1), 77–103 (1997)
18. Spohn, W.: Ordinal conditional functions: a dynamic theory of epistemic states. In: Harper, W.L., Skyrms, B. (eds.) Causation in Decision, Belief Change, and Statistics, vol. II, pp. 105–134. Kluwer Academic Publishers, Dordrecht (1988)
19. Weydert, E.: System JZ - How to build a canonical ranking model of a default knowledge base. In: Proceedings KR'98. Morgan Kaufmann, San Mateo (1998)
20. Wielemaker, J., Schrijvers, T., Triska, M., Lager, T.: SWI-Prolog. CoRR, abs/1011.5332, (2010) (to appear in Theory and Practice of Logic Programming)