# Stepwise Optimization of a Constraint Logic Program for the Computation of Ranking Functions

Christoph Beierle[1], Gabriele Kern-Isberner[2], Karl Södler[1]

[1]Dept. of Computer Science, FernUniversität in Hagen, 58084 Hagen, Germany
[2]Dept. of Computer Science, TU Dortmund, 44221 Dortmund, Germany

**Abstract:** Ordinal conditional functions (OCFs) can be used for assigning a semantics to qualitative conditionals of the form *if $A$ then (normally) $B$*. The set of OCFs accepting all conditionals in a knowledge base $\mathcal{R}$ can be specified as the solutions of a constraint satisfaction problem $CR(\mathcal{R})$. In this paper, we present three optimizations of a high-level, declarative CLP program solving $CR(\mathcal{R})$ and illustrate the benefits of these optimizations by various examples.

## 1 Introduction

Common sense knowledge like "*birds are animals*" or "*birds fly*" can be expressed as conditionals of the form "*if $A$ then (normally) $B$*", formally denoted by $(B|A)$. Such conditionals are very different from material implications $A \Rightarrow B$, and their semantics is often given by what can be seen as an epistemic state of an intelligent agent. In this paper, we consider Spohn's ordinal conditional functions (OCF) [Spo88] providing a semantics for qualitative conditionals. Given a knowledge base $\mathcal{R} = \{R_1, \ldots, R_n\}$ of conditionals, we are interested in determining all OCFs that on the one hand accept $\mathcal{R}$, and that on the other hand, are as unspecific as possible. While for system Z [GMP93, GP96] a unique most unspecific OCF exists, here we consider the framework of c-representations [KI02] that has been developed as a qualitative realization of the ideas underlying the probabilistic maximum entropy principle [Par94, PV97, KI98] and provides a generalization of system Z* [GMP93].

In [BKI11], as a challenge for constraint programming, the set of all c-representations accepting $\mathcal{R}$ is specified via the set of solution vectors of a finite domain constraint satisfaction problem $CR(\mathcal{R})$. For computing the solutions of $CR(\mathcal{R})$, in [BKS11, BKS12] a high-level, declarative CLP program `GenOCF` is developed. In this paper, we extend the work in [BKS11, BKS12] by developing three optimizations $CR(\mathcal{R})$ and `GenOCF`:

- We optimize $CR(\mathcal{R})$ by replacing inequations in $CR(\mathcal{R})$ by equations such that no minimal solutions are lost (Sec. 4). This optimizations is motivated by the fact that one is mainly interested in minimal solutions of $CR(\mathcal{R})$.

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

- We optimize `GenOCF` by simplifying minimum constraints over a sum of terms containing a 0 (Sec. 5). This optimization is possible since all solution values are non-negative.
- Finally, we further reduce the number of minimum constraints over a sum of terms by considering multiple terms and subterms (Sec. 6).

Before presenting the optimizations in detail, we recall the OCF background (Sec. 2) and the the basics of $CR(\mathcal{R})$ and `GenOCF` (Sec. 3) as far as is needed here. In Section 7, a first implementation of the different optimizations are evluated with respect to various examples, and Section 8 concludes and points out further work.

## 2 Background

We start with a propositional language $\mathcal{L}$, generated by a finite set $\Sigma$ 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 $\Omega$ denote the set of possible worlds over $\mathcal{L}$; $\Omega$ 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 $\Sigma$. For $\omega \in \Omega$, $\omega \models A$ means that the propositional formula $A \in \mathcal{L}$ holds in the possible world $\omega$.

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 $\Omega$ 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 [DeF74] (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*) [Spo88], and *possibility distributions* [BDP92], assigning degrees of plausibility, or of possibility, respectively, to

| $\omega$ | $\kappa_1(\omega)$ | $\kappa_2(\omega)$ | $\kappa_3(\omega)$ |
|---|---|---|---|
| $fba$ | 0 | 0 | 0 |
| $fb\overline{a}$ | 1 | 1 | 1 |
| $f\overline{b}a$ | 0 | 0 | 0 |
| $f\overline{b}\,\overline{a}$ | 0 | 0 | 0 |

| $\omega$ | $\kappa_1(\omega)$ | $\kappa_2(\omega)$ | $\kappa_3(\omega)$ |
|---|---|---|---|
| $\overline{f}ba$ | 1 | 1 | 2 |
| $\overline{f}b\overline{a}$ | 1 | 2 | 1 |
| $\overline{f}\,\overline{b}a$ | 0 | 0 | 0 |
| $\overline{f}\,\overline{b}\,\overline{a}$ | 0 | 0 | 0 |

Figure 1: Different ranking function $\kappa_i$ accepting the rule set $\mathcal{R}_{birds}$ given in Example 1

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 [Spo88]. 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 $\kappa$ uniquely extends to a function (also denoted by $\kappa$) mapping sentences and rules to $\mathbb{N} \cup \{\infty\}$ and being defined by

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

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

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

for conditionals $(B|A) \in (\mathcal{L} \mid \mathcal{L})$. Note that $\kappa((B|A)) \geqslant 0$ since any $\omega$ satisfying $AB$ also satisfies $A$ and therefore $\kappa(AB) \geqslant \kappa(A)$. The belief of an agent being in epistemic state $\kappa$ with respect to a default rule $(B|A)$ is determined by the satisfaction relation $\models_{\mathcal{O}}$ given by:

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

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

Given a knowledge base $\mathcal{R} = \{R_1, \ldots, R_n\}$ of conditionals, a ranking function $\kappa$ that accepts every $R_i$ repesents an epistemic state of an agent accepting $\mathcal{R}$. If there is no $\kappa$ 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 $\kappa$ accepting $\mathcal{R}$, each representing a complete set of beliefs with respect to every possible formula $A$ and every conditional $(B|A)$.

**Example 1** Let $\mathcal{R}_{birds} = \{R_1,\ R_2,\ R_3\}$ be the following set of conditionals:

$$\begin{array}{lll} R_1: & (f|b) & \underline{b}\textit{irds }\underline{f}\textit{ly} \\ R_2: & (a|b) & \underline{b}\textit{irds are }\underline{a}\textit{nimals} \\ R_3: & (a|fb) & \underline{f}\textit{lying }\underline{b}\textit{irds are }\underline{a}\textit{nimals} \end{array}$$

In Figure 1, three different OCFs $\kappa_1$, $\kappa_2$, $\kappa_3$ accepting $\mathcal{R}_{birds}$ are given. Thus, for any $i \in \{1,2,3\}$ and $j \in \{1,2,3\}$ it holds that $\kappa_i \models_{\mathcal{O}} R_j$. In order to ilustrate the evaluation of beliefs, consider the conditional $(a|b\overline{f})$ (*"Are non-flying birds animals?"*) that is not contained in $\mathcal{R}$. For $\kappa_3$, we get $\kappa_3(ab\overline{f}) = 2$ and $\kappa_3(\overline{a}b\overline{f}) = 1$ and therefore $\kappa_3 \not\models_{\mathcal{O}} (a|b\overline{f})$ so that the conditional $(a|b\overline{f})$ is not accepted by $\kappa_3$. On the other hand, for $\kappa_2$ we get $\kappa_2(ab\overline{f}) = 1$ and $\kappa_2(\overline{a}b\overline{f}) = 2$ and therefore $\kappa_2 \models_{\mathcal{O}} (a|b\overline{f})$.

The full beliefs about non-flying birds being animals or not, represented by the conditionals $(a|b\overline{f})$ and $(\overline{a}|b\overline{f})$, are given by the following table:

$$\begin{array}{lll} \kappa_1 \not\models_{\mathcal{O}} (a|b\overline{f}) & \kappa_2 \models_{\mathcal{O}} (a|b\overline{f}) & \kappa_3 \not\models_{\mathcal{O}} (a|b\overline{f}) \\ \kappa_1 \not\models_{\mathcal{O}} (\overline{a}|b\overline{f}) & \kappa_2 \not\models_{\mathcal{O}} (\overline{a}|b\overline{f}) & \kappa_3 \models_{\mathcal{O}} (\overline{a}|b\overline{f}) \end{array}$$

An agent being in epistemic state $\kappa_2$ believes that non-flying birds are animals and does not believe that non-flying birds are not animals. An agent being in epistemic state $\kappa_3$ does not believe that non-flying birds are animals and believes that non-flying birds are not animals. An agent being in epistemic state $\kappa_1$ is completely indifferent with respect to non-flying birds being animals or not, since she considers a world where non-flying birds are animals as equally plausible (or equally surprising) as a world where non-flying birds are not animals. ∎

Thus, Example 1 illustrates that every OCF $\kappa$ accepting a knowledge base $\mathcal{R}$ inductively completes the knowledge given by $\mathcal{R}$, and that in general there are different ways of completing the knowledge. In principle, one is interested in characterizing and determining the full set of accepting OCFs, and it is a crucial question whether some $\kappa$ is to be preferred to some other $\kappa'$, or whether among the preferred ones there is a unique "best" $\kappa$.

# 3 Computing OCFs as Solutions of a Constraint Satisfaction Problem

## 3.1 The Constraint Satisfaction Problem $CR(\mathcal{R})$

Different ways of determining a ranking function for a knowledge base $\mathcal{R}$ are given by *system Z* [GMP93, GP96] or its more sophisticated extension *system Z*$^*$ [GMP93], see also [BP99]; for an approach using rational world rankings see [Wey98]. 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 [Par94, PV97, KI98]; 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 [KI02] a generalization of system Z$^*$ is suggested. Based on an algebraic treatment of conditionals, the notion of *conditional indifference* of $\kappa$ with respect to $\mathcal{R}$ is defined and the following criterion for conditional indifference is given: An OCF $\kappa$ 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 $\omega$ has rank 0), for each rule $(B_i|A_i)$ the values $\kappa_i^+, \kappa_i^-$ determine how the rank of each satisfying world and of each falsifying world, respectively, should be changed:

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

$\kappa_0$ 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* [KI02, KI01]: Any ranking function $\kappa$ 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

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

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 $\kappa$ 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 [BKS12] 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 formulation of $CR(\mathcal{R})$ requires that the $\kappa_i^-$ are natural numbers (and not just rational numbers) and that $\min(\emptyset) = \infty$.

**Definition 2 [$CR(\mathcal{R})$ [BKS12]]** 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, for all $i \in \{1, \ldots, n\}$:

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

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

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

**Proposition 3** ***[BKS12]*** *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 $\kappa$ defined by (9) and denoted by $\kappa_{\vec{\kappa}}$, is an OCF that accepts $\mathcal{R}$:*

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

**Example 4** Let $\mathcal{R}_{birds} = \{R_1,\ R_2,\ R_3\}$ be as in Example 1. From (8) we get

$$\begin{array}{l} \kappa_1^- > 0, \\ \kappa_2^- > 0 - min\{\kappa_1^-,\ \kappa_3^-\}, \\ \kappa_3^- > 0 - \kappa_2^- \end{array}$$

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

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

are two different solutions of $CR(\mathcal{R}_{birds})$. The OCF $\kappa_{sol_1}$ (resp. $\kappa_{sol_2}$) induced by $sol_1$ (resp. $sol_2$) according to (9) is $\kappa_1$ (resp. $\kappa_2$) as given in Example 1. ∎

### 3.2 The CLP program `GenOCF`

In `GenOCF` [BKS12] a knowledge base $\mathcal{R} = \{(B_1|A_1), \ldots, (B_n|A_n)\}$ is given as a Prolog code file providing the predicates `variables/1`, `indices/1` and `conditional/3`. 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 $\Sigma$ given by the predicate `variables([`$a_1$`,...,`$a_m$`])`. The fixed index ordering given 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 $\kappa^-$-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 $\Sigma$ 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 $\dot{a}_k = a_k$ if $b_k = \mathtt{1}$, $\dot{a}_k = \overline{a_k}$ if $b_k = \mathtt{0}$, and $\dot{a}_k = \top$ if $b_k = \_$.

**Example 5** The internal representation of the knowledge base $\mathcal{R}_{birds}$ from Ex. 1 is:

```
variables([f,b,a]).                          % propositional variables
indices([1,2,3]).                            % list of indices for
                                             % conditionals
%               f b a     f b a
conditional(1,[[_,1,_]],[[1,_,_]]).  % (f|b)  birds fly
conditional(2,[[_,1,_]],[[_,_,1]]).  % (b|p)  birds are animals
conditional(3,[[1,1,_]],[[_,_,1]]).  % (-f|p) flying birds are
                                             %        animals
```

■

Furthermore, `GenOCF` provides the predicates

```
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
```

realising the evaluation of the indicator function (1) given in Sec. 2. The remaining part of `GenOCF` needed for $\texttt{GenOCF}_O$ is the determination of vectors $\vec{\kappa}$ as solutions of $CR(\mathcal{R})$. The corresponding `GenOCF` predicates closely follow the abstract specification of $CR(\mathcal{R})$ developed in Section 3.1 and are given in Figure 2.

## 4 Optimizing $CR(\mathcal{R})$ by Transforming Inequations to Equations

The constraints in $CR(\mathcal{R})$ given by (8) ensure that each conditional $(B_i|A_i) \in \mathcal{R}$ is accepted. Since all $\kappa_i^-$ are assumed to be natural numbers, we could replace the strict inequation (8) by

$$\kappa_i^- \geqslant 1 + \min_{\omega \models A_i B_i} \sum_{\substack{j \neq i \\ \omega \models A_j \overline{B_j}}} \kappa_j^- - \min_{\omega \models A_i \overline{B}_i} \sum_{\substack{j \neq i \\ \omega \models A_j \overline{B_j}}} \kappa_j^- \tag{10}$$

without changing the set of solutions $Sol_{CR}(\mathcal{R})$. Generally, one is interested in minimal solutions, so one could be tempted so for minimizing the values of all $\kappa_i^-$, one could be tempted to replace the non-strict inequality in (10) by an equality as in:

$$\kappa_i^- = 1 + \min_{\omega \models A_i B_i} \sum_{\substack{j \neq i \\ \omega \models A_j \overline{B_j}}} \kappa_j^- - \min_{\omega \models A_i \overline{B}_i} \sum_{\substack{j \neq i \\ \omega \models A_j \overline{B_j}}} \kappa_j^- \tag{11}$$

However, using just (11), one might loose a solution in the case where the right hand side of the inequation (8) is negative since then (11) would require that $\kappa_i^-$ is negative. If (11) does not hold, the values of the vector $(\kappa_i^-, \ldots, \kappa_n^-)$ ensure that the conditional $(B_i|A_i)$ is accepted, so that in that case no additional requirement on $\kappa_i^-$ is needed.

Thus, we can use (11) only if

$$1 + \min_{\omega \models A_i B_i} \sum_{\substack{j \neq i \\ \omega \models A_j \overline{B_j}}} \kappa_j^- \geqslant \min_{\omega \models A_i \overline{B}_i} \sum_{\substack{j \neq i \\ \omega \models A_j \overline{B_j}}} \kappa_j^- \tag{12}$$

```
kappa(KB, K) :-       % K is kappa vector of c-representation
  consult(KB),        %    for 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),       % gen. K = [Kappa_1,...,Kappa_N] of free var.
  domain(K, 0, N),    % 0 <= kappa_I <= N for all I accord. to (7)
  constrain_K(Is, K). % generate constraints according to (8)

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) :-          % gen. constr. for kappa_I as in (8)
  verifying_worlds(I, VWs),   % all worlds verif. I-th cond.
  falsifying_worlds(I, FWs),  % all worlds falsif. I-th cond.
  list_of_sums(I, K, VWs, VS), % VS list of sums for verif. worlds
  list_of_sums(I, K, FWs, FS), % FS list of sums for falsif. worlds
  minimum(Vmin, VS),          % Vmin minium for verif. worlds
  minimum(Fmin, FS),          % Fmin minium for falsif. worlds
  element(I, K, Ki),          % Ki constraint variable for kappa_I
  Ki #> Vmin - Fmin.          % constraint for kappa_I as in (8)

% 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).
```

Figure 2: CLP program `GenOCF` determining vectors as solutions of $CR(\mathcal{R})$

```
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) :-              % gen. constr. for kappa_I as in (13)
  verifying_sumlists(I,K,VS),     % gen. list of sums of verif. worlds
  falsifying_sumlists(I,K,FS),    % gen. list of sums of 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)
% ================================================
% BEGIN      equation
% ================================================
  FSmin is Vmin + 1,              % Fmin may not be greater than Vmin+1,
  minimum(Fmin, [FSmin|FS]),      % Fmin minium for falsifying worlds
  element(I, K, Ki),              % Ki constraint variable for kappa_I
  Ki #= 1 + Vmin - Fmin.          % constraint for kappa_I as in (13)
                                  %   as equation
% ================================================
% END        equation
% ================================================
```

Figure 3: Optimization of GenOCF by replacing inequations by equations

holds. Putting these constraints together yields

$$\kappa_i^- = 1 + \min_{\omega \models A_i B_i} \sum_{\substack{j \neq i \\ \omega \models A_j \overline{B_j}}} \kappa_j^- - \min\{1 + \min_{\omega \models A_i B_i} \sum_{\substack{j \neq i \\ \omega \models A_j \overline{B_j}}} \kappa_j^-, \quad \min_{\omega \models A_i \overline{B}_i} \sum_{\substack{j \neq i \\ \omega \models A_j \overline{B_j}}} \kappa_j^-\} \tag{13}$$

Note that just as required, (13) reduces to $\kappa_i^- = 0$ if (12) holds. Thus, our first optimization of $CR(\mathcal{R})$ is to replace (8) by (13). Note that this optimization transforms a strictly-less-than relationship into an equation; thus it should be clearly distinguished from the modelling of a constraint $x < y$ by $x + 1 \leqslant y$ which might be done by the underlying constraint solver.

For different notions of minimality, resulting from different orderings of the solution vectors $\vec{\kappa}$ in $Sol_{CR}(\mathcal{R})$, this optimization does not loose any minimal solution. For instance, this observation holds when $\vec{\kappa}$ is compared to $\vec{\kappa}'$ by summing up the elements of the vectors, by a componentwise ordering of the vectors, or by a componentwise ordering of the full ranking functions induced by the vectors according to (9) (cf. [BKS12]).

A declarative implementation of this optimization in GenOCF is obtained by replacing the last three subgoals of in the predicate constrain_Ki/2 (cf. Figure 2) is shown in Figure 3. Note that the new program code in Figure 3 closely follows the formulas given in (13).

It turned out that this optimization had a significant effect on the runtime of GenOCF as

will be illustrated in Section 7.

## 5 Optimize Minimum Constraint Over List of Sums Containing 0

A typical set of constraints obtained from (8) may contain

$$\kappa_1^- > min\{\kappa_2^- + \kappa_3^-, 0\} - min\{\kappa_2^-, 0\} \tag{14}$$

Since all $\kappa_i^-$ are non-negative, both minimum constraints of this inequation will always return 0 regardless of any other constraints. Thus, we can optimise both the first minimum constraint $min\{\kappa_2^- + \kappa_3^-, 0\}$ and the second minimum constraint $min\{\kappa_2^-, 0\}$ in (14) by 0, and further evaluating $0 * 0$ to 0 transforms (14) into just

$$\kappa_1^- > 0 \tag{15}$$

The optimisation is to avoid superfluous sum constraints where the minimum over a list of sums can be replaced by 0. Indead, this optimization can always be applied for any consistent set of conditionals $\mathcal{R}$. If a knowledge base is splitted into *ordered partitions* [GP96], all conditionals assigned to the first ordered partition give raise to this optimization. The tolerance condition [GP96] ensures that a verifying world for each conditional of the first orderd partition exists such that no other conditional of the whole knowledge base $\mathcal{R}$ is falsified by this world. This leads to a sum of 0 as shown above.

To realise this optimisation, the predicate `constrain_Ki/2` (cf. Figure 2) must be modified:

- It must be checked whether a verifying world exists such that no other conditional is falsified by that world

  ```
  ((once((verify(I,World), \+ falsify(_,World))))
  ```

  and dually, if such a falsifying World exists:

  ```
  ((once((falsify(I,World), \+ (falsify(J,World),J \= I))))
  ```

- In case of such a world could be found, a list with only one constraint to sum 0 will be created. Otherwise the original predicates `verifying_worlds/2` (as well as `falsifying_worlds/2` and `list_of_sums/4` will be used to create the list of sum constraints.

The modification is implemented using two new predicates `verifying_sumlists/3` and `falsifying_sumlists/3`. Calls to these new prediactes replace the calls to `verifying_worlds/2` and `falsifying_worlds/2` in `constrain_Ki/2`. The optimized version of `constrain_Ki/2` that also takes into account the first optimization presented in Section 4 is given in Figure 4 and 5.

```
constrain_K([],_).                  % generate constraints for
constrain_K([I|Is],K) :-            % all kappa_I as in (7)
  constrain_Ki(I,K), constrain_K(Is,K).

constrain_Ki(I,K) :-              % gen. constr. for kappa_I as in (7)
  verifying_worlds(I, VWs),     % all worlds verif. I-th conditional
  falsifying_worlds(I, FWs),    % all worlds falsif. I-th conditional
  list_of_sums(I, K, VWs, VS), % VS list of sums for verif. worlds
  list_of_sums(I, K, FWs, FS), % FS list of sums for falsif. worlds
% ================================================
% BEGIN      Sum Constraint 0
% ================================================
  verifying_sumlists(I,K,VS),  % gen. list of sums of verif. worlds
  falsifying_sumlists(I,K,FS), % gen. list of sums of falsif. worlds
% ================================================
% END        Sum Constraint 0
% ================================================
  FSmin is Vmin + 1,            % Fmin may never be greater than
                                %   Vmin + 1, otherwise Ki would
                                %   decrease below 0
  minimum(Fmin, [FSmin|FS]),    % Fmin minium for falsifying worlds
  element(I, K, Ki),            % Ki constr. variable for kappa_I
  Ki #= 1 + Vmin - Fmin.        % constraint for kappa_I as in (7)
                                %   as eqution

% ================================================
% BEGIN      Sum Constraint 0
%      minimum([9,2,8,0,8], Min) replaced with Min #= 0
% ================================================
% Generates all lists of sums as in (7) of verifying worlds
% Includes the search for verifying worlds.
% Opt71: If a verifying world does not falsify another conditional
% a minimum of 0 of all sums is assumed.
%     I   index from 1,...,N
%     K   kappa vector
%     VSumSlist  list of sums:
verifying_sumlists(I,K,VSumList) :-
  ((once((verify(I,World), \+ falsify(_,World)))) % does a
                                % verifying world exist which does
                                % not falsify another conditional?
      ->  VSumList = [0]        % Assume minimum as 0
      ;  verifying_worlds(I, Worlds),    % List of all verif. worlds
         list_of_sums(I,Worlds,K,VSumList) % List of Sums
      ).
```

Figure 4: Optimization of `GenOCF` w.r.t the minimum of list of sums (part 1/2)

## 6 Reduce Number of Sum Constraints

Suppose that from the constraints in $CR(\mathcal{R})$ we have obtained the constraint

$$\begin{aligned}\kappa_2^- > min\{0,\, \kappa_6^- + \kappa_7^-,\, \kappa_4^- + \kappa_5^-,\, \kappa_4^- + \kappa_5^- + \kappa_6^-,\, \kappa_1^-,\, \kappa_1^-,\, \kappa_1^-,\, \kappa_1^-\} \\ - min\{\kappa_3^-,\, \kappa_3^-,\, \kappa_3^- + \kappa_4^-,\, \kappa_3^- + \kappa_4^-,\, \kappa_1^-,\, \kappa_1^-,\, \kappa_1^-,\, \kappa_1^-\} \end{aligned} \tag{16}$$

```
% Same as verifying_sumlist but starting with falsifying worlds.
falsifying_sumlists(I,K,VSumList) :-
       ((once((falsify(I,World), \+ (falsify(J,World),J \= I))))
                                     % does a falsifying world exist
                                     % falsifying ANOTHER (J \= I)
                                     % conditional?
       →  VSumList = [0]
       ;  falsifying_worlds(I, Worlds),
          list_of_sums(I,Worlds,K,VSumList)
       ).
% ================================================
% END         Sum Constraint 0
% ================================================
```

Figure 5: Optimization of GenOCF w.r.t the minimum of list of sums (part 2/2)

(In fact, this constraint arises for one of the examples given in [Mül04]). With the optimization given in the previous section, (16) can be replaced by

$$\kappa_2^- > min\{0\} - min\{\kappa_3^-,\ \kappa_3^-,\ \kappa_3^- + \kappa_4^-,\ \kappa_3^- + \kappa_4^-,\ \kappa_1^-,\ \kappa_1^-,\ \kappa_1^-,\ \kappa_1^-\} \tag{17}$$

A further simplification is to remove duplicate sum constraints, yielding

$$\kappa_2^- > min\{0\} - min\{\kappa_3^-,\ \kappa_3^- + \kappa_4^-,\ \kappa_1^-\} \tag{18}$$

Furthermore, it is evident that a minimum constraint $min\{\kappa_3^-,\ \kappa_3^- + \kappa_4^-\}$ may be reduced to $min\{\kappa_3^-\}$ since $\kappa_i^- \geqslant 0$. So if all constraint variables of a sum constraint are also part of another sum constrain, the larger sum constraint is superfluous and does not have to be issued to the constraint solver. Thus, our third optimization is to avoid

- duplicate sum constraints
- sum constraints greater or equal to other sum constraints

in order not to post a big bulk of redundant constraints, speeding up GenOCF before the constraint solver starts to calculate solutions and aiming at improving the overall performance of GenOCF.

Based on the optimization in the previous sections, the new predicates verifying_sumlists/3 and falsifying_sumlists/3 will be modified.

Generally, the way to to avoid spare sum constraints and to identify only relevant sum constraints will be changed. Two new predicates verify_falsify_stripped/2 and falsify_falsify_stripped/2 will replace the predicates verifying_worlds/2 and falsifying_worlds/2. With these new predicates the search for verifying and falsifying worlds changes: The search for verifying resp. falsifying worlds is defined by constraints, so the iteration to search falsifying worlds for each verifying world is obsolet (see predicate falsifying_except/3). With predicate stripedList all sum constraints greater or equal to other sum constraints are removed. The complete implementation of this optimization is given in Figures 6, 7, and 8.

```
% Generates all lists of sums as in (8) of verifying worlds
% Includes the search for verifying worlds.
% Opt71: If a verifying world does not falsify another conditional
% a minimum of 0 of all sums is assumed.
%     I   index from 1,...,N
%     K   kappa vector
%     VSumSlist  list of sums:
verifying_sumlists(I,K,VSumList) :-
  ((once((verify(I,World), \+ falsify(_,World)))) % does a
                                % verifying world exist which does
                                % not falsify another conditional?
       →  VSumList = [0]        % Assume minimum as 0
       ;  verifying_worlds(I, Worlds),  % List of all verif. worlds
          list_of_sums(I,Worlds,K,VSumList) % List of Sums
% ================================================
% BEGIN      Bulk Constraints
% ================================================
             verify_falsify_stripped(I,Is),
             list_of_sums(Is,K,VSumList).
% ================================================
% END        Bulk Constraints
% ================================================
       ).

% Same as verifying_sumlist but starting with falsifying worlds.
falsifying_sumlists(I,K,VSumList) :-
       ((once((falsify(I,World), \+ (falsify(J,World),J \= I))))
                                % does a falsifying world exist
                                % falsifying ANOTHER (J \= I)
                                % conditional?
       ->     VSumList = [0]
       ;      falsifying_worlds(I, Worlds),
              list_of_sums(I,Worlds,K,VSumList)
% ================================================
% BEGIN      Bulk Constraints
% ================================================
             falsify_falsify_stripped(I,Is),
             list_of_sums(Is,K,VSumList).
% ================================================
% END        Bulk Constraints
% ================================================
       ).

% ================================================
% BEGIN      Bulk Constraints
% ================================================
% Creates list of kappa minus indices to create the list of sum
% constraints for the FIRST minimisation of the c-representation (8)
% - no duplicate sum constraints
% - striped list
%   i.e given   [[1,2],[1,2,3],[4,5],[4,5,6],[6,7]]
%       striped [[1,2],[4,5],[6,7]]
```

Figure 6: Optimization of GenOCF by reducing sum constraints (part 1/3)

```
% in/output
% F    I      Index of that conditional which is excluded
%               from the search
% Is   O      Striped list of list of indices of kappa minus
%               variables
verify_falsify_stripped(V,Is) :-
       findall(A1,setof(A,(verify(V,W),falsify_except(A,V,W)),A1),A2),
                                  % Raw list with duplicates
       setof(A3,member(A3,A2),A4),  % Remove duplicates
       stripedList(A4,Is).          % Strip spare sum constraints

% Creates list of kappa minus indices to create the list of sum
% constraints for the SECOND minimisation of the c-representation (8)
% - no duplicate sum constraints
% - striped list
%   i.e given   [[1,2],[1,2,3],[4,5],[4,5,6],[6,7]]
%       striped [[1,2],[4,5],[6,7]]
% in/output
% F    I      Index of that conditional which is excluded
%               from the search
% Is   O      Striped list of list of indices of kappa minus
%               variables
falsify_falsify_stripped(F,Is) :-
    findall(A1,setof(A,(falsify(F,W),falsify_except(A,F,W)),A1),A2),
                               % Raw list with duplicates
    setof(A3,member(A3,A2),A4),  % Remove duplicates
    stripedList(A4,Is).          % Strip spare sum constraints

% Seach world W falsifying conditional represented by index I
% except conditional represented by X
% Input / Output
% I    I / O  Index of a falsified conditional
% X    I      Index of an excluded conditional
% W    I / O  World falsifying conditional I
falsify_except(I,X,W) :-
    variables(Vs),  % Signatur / Variables of the knowledge base
    length(Vs,NVs), % N = Number of variables
    length(W,NVs),  % Define constr. variables (search for worlds)
    domain(W,0,1),  % Initialize constr. variables / define domain
    indices(Cs),        % Indices of conditionals
    length(Cs,NCs),     % Number of indices of conditionals
    domain([I],1,NCs),  % Initialise domain of constraint variable i
    I #\= X,            % Exclude conditional X
    conditional(I,P,C), % Get conditional specification
    member(Pc,P),       % extract precondition
    member(Cc,C),       % extract consequence
    conj(Pc,W),         % constraints: Search models of the
                        %              precondition
    neg_conj(Cc,W),     % constraints: Search models of negated
                        %              condition (falsify)
    labeling([],W),     % Execute search for worlds
    labeling([],[I]).   % Execute search for conditional indices

% Stripe a list of sum constraints (here kappa minus indices)
%   i.e given   [[1,2],[1,2,3],[4,5],[4,5,6],[6,7]]
%       striped [[1,2],[4,5],[6,7]]
```

Figure 7: Optimization of `GenOCF` by reducing sum constraints (part 2/3)

```
% input
%      E      List without duplicates
%      USL    Unstriped List
% Output
%      SL     Striped List
stripedList(E,SL) :- stripedList(E,E,SL).
stripedList([],_,[]).
stripedList([E|Es],USL,SL) :-
    stripedList(Es,USL,SLp),
    ((once((member(L,USL), E \= L, inList(L,E))))
                % Is there a L in USL such that L is a subset of E?
          →  SL = SLp            % YES -> stripe E out
          ;  append([E],SLp,SL)  % NO  -> add E to the striped list
    ).

% Checks, whether list A is a subset of list B:
% i.e.: A = [1,2] is a subet of B = [0,1,2,3]
% Input
%      [H|T]  List A
%      L      List B
inList([],_).
inList([H|T],L) :-
       inList(T,L),
       member(H,L).

% Create the sum constraints along
% a list of List of indices of kappa minus
% Input
%   [Is|IIs] list of list of indices
%   K        kappa minus constraint variables
% Output
%   FsumList List of sum constraints
list_of_sums(_,[],_,[]).
list_of_sums(I,[W|Ws],K,FSumList) :-
   list_of_sums(I,Ws,K,FSumTmp),
   findall(F,(falsify(F,W), F \= I),Fs),
   sum_kappa(Fs,K,Sum),
   append([Sum],FSumTmp,FSumList).
list_of_sums([],_,[]).
list_of_sums([Is|IIs],K,FSumList) :-
   list_of_sums(IIs,K,FSumTmp),
   sum_kappa(Is,K,Sum),
   append([Sum],FSumTmp,FSumList).

% ================================================
% END        Bulk Constraints
% ================================================
```

Figure 8: Optimization of `GenOCF` by reducing sum constraints (part 3/3)

# 7 Implementation and First Evaluation

We implemented the three optimizations developed above and applied them to the sythetic knowledge bases of the form `kb_synth<`$n$`>_c<`$2n$–1`>.pl` which were introduced in [BKS12]. In Figures 9 and 10, results of applying `GenOCF` with and without the optimizations are shown. It is intesting to note that the first optimizations has the biggest effect on the runtimes, while the other two optimizations are still significant.

| Knowlegde Base | #Var | #Cond | Runtime | **Opt. 1** | **Opt. 2** | **Opt. 3** | **Program** | constrain | labeling |
|---|---|---|---|---|---|---|---|---|---|
| KB/kb_synth4_c7.pl | 5 | 7 | sicstus | N | N | N | 3430 | 40 | 3390 |
| KB/kb_synth4_c7.pl | 5 | 7 | sicstus | Y | N | N | 2380 | 30 | 2350 |
| KB/kb_synth4_c7.pl | 5 | 7 | sicstus | Y | Y | N | 810 | 30 | 780 |
| KB/kb_synth4_c7.pl | 5 | 7 | sicstus | Y | Y | Y | 30 | 30 | 0 |
| | | | | | | | | | |
| KB/kb_synth5_c6.pl | 6 | 6 | sicstus | N | N | N | 11070 | 50 | 11020 |
| KB/kb_synth5_c6.pl | 6 | 6 | sicstus | Y | N | N | 2920 | 30 | 2890 |
| KB/kb_synth5_c6.pl | 6 | 6 | sicstus | Y | Y | N | 670 | 20 | 650 |
| KB/kb_synth5_c6.pl | 6 | 6 | sicstus | Y | Y | Y | 10 | 10 | 0 |

Figure 9: Runtimes of `GenOCF` with and without employing optimizations 1, 2,3

| **Knowledge Base** | | | **runtime (milliseconds)** | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | **Optimization 1** | | **Optimization 1, 2** | | **Optimization 1, 2, 3** | |
| | **#Var** | **#Cond** | **program** | **constrain** | **program** | **constrain** | **program** | **constrain** |
| kb_synth9_c1.pl | 10 | 1 | **290** | *290* | **70** | *60* | **100** | *70* |
| kb_synth9_c2.pl | 10 | 2 | **220** | *220* | **120** | *100* | **110** | *110* |
| kb_synth9_c3.pl | 10 | 3 | **380** | *370* | **150** | *130* | **200** | *190* |
| kb_synth9_c4.pl | 10 | 4 | **930** | *920* | **170** | *170* | **170** | *160* |
| kb_synth9_c5.pl | 10 | 5 | **15060** | *15060* | **200** | *200* | **270** | *270* |
| kb_synth9_c6.pl | 10 | 6 | **2410** | *2410* | **250** | *230* | **300** | *300* |
| kb_synth9_c7.pl | 10 | 7 | **11250** | *11240* | **280** | *270* | **320** | *310* |
| kb_synth9_c8.pl | 10 | 8 | **4590** | *4580* | **310** | *300* | **350** | *340* |
| kb_synth9_c9.pl | 10 | 9 | **4810** | *4790* | **350** | *330* | **380** | *370* |
| kb_synth9_c10.pl | 10 | 10 | **6820** | *6800* | **660** | *660* | **660** | *650* |
| kb_synth9_c11.pl | 10 | 11 | **7550** | *7490* | **1240** | *1220* | **1140** | *1140* |
| kb_synth9_c12.pl | 10 | 12 | **5370** | *5350* | **1890** | *1880* | **1600** | *1600* |
| kb_synth9_c13.pl | 10 | 13 | **36860** | *36850* | **4130** | *4120* | **2130** | *2130* |
| kb_synth9_c14.pl | 10 | 14 | **6030** | *6020* | **3770** | *3750* | **2730** | *2720* |
| kb_synth9_c15.pl | 10 | 15 | **26090** | *26080* | **5380** | *5380* | **3430** | *3320* |
| kb_synth9_c16.pl | 10 | 16 | **22640** | *22630* | **6740** | *6720* | **4260** | *4240* |
| kb_synth9_c17.pl | 10 | 17 | **22510** | *22510* | **20080** | *20080* | **5550** | *5540* |

Figure 10: Runtimes of `GenOCF` with optimizations for knowldege bases with 10 variables and increasing number of conditionals

## 8 Conclusions and Further Work

We presented three stepwise optimizations of a high-level declarative CLP program computing the solutions of a constraint satisfaction problem specifying ranking functions that accept the conditionals of a knowlegde base. A first implementation and its applications to various examples illustrates the effects of the optimizations. The biggest effect is obtained by a sharpening of the original constraint satisfaction problem; this optimizations transforms inequations to equations, but without loosing any minimal solutions. Among our current work is the development of further refinements and a more detailed investigation of the benefits of the different optimizations.

## References


[BDP92] S. Benferhat, D. Dubois, and H. Prade. Representing default rules in possibilistic logic. In *Proceedings 3th International Conference on Principles of Knowledge Representation and Reasoning KR'92*, pages 673–684, 1992.

[BKI11] C. Beierle and G. Kern-Isberner. On the Computation of Ranking Functions for Default Rules – A Challenge for Constraint Programming. In *Proc. Deklarative Modellierung und effiziente Optimierung mit Constraint-Technologie. Workshop at GI Jahrestagung 2011*, 2011.

[BKS11] C. Beierle, G. Kern-Isberner, and K. Södler. A Constraint Logic Programming Approach for Computing Ordinal Conditional Functions. In *25th Workshop on Logic Programming (WLP 2011)*, INFSYS Research Report 1843-11-06, pages 9–21. TU Wien, 2011.

[BKS12] C. Beierle, G. Kern-Isberner, and K. Södler. A Declarative Approach for Computing Ordinal Conditional Functions Using Constraint Logic Programming. In *19th International Conference on Applications of Declarative Programming and Knowledge Management, INAP 2011, and 25th Workshop on Logic Programming, WLP 2011, Wien, Austria, Revised Selected Papers*, Lecture Notes in Artificial Intelligence. Springer, 2012. (to appear).

[BP99] R.A. Bourne and S. Parsons. Maximum entropy and variable strength defaults. In *Proceedings Sixteenth International Joint Conference on Artificial Intelligence, IJCAI'99*, pages 50–55, 1999.

[DeF74] B. DeFinetti. *Theory of Probability*, volume 1,2. John Wiley & Sons, 1974.

[GMP93] M. Goldszmidt, P. Morris, and J. Pearl. A maximum entropy approach to nonmonotonic reasoning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(3):220–232, 1993.

[GP96] M. Goldszmidt and J. Pearl. Qualitative probabilities for default reasoning, belief revision, and causal modeling. *Artificial Intelligence*, 84:57–112, 1996.

[KI98] G. Kern-Isberner. Characterizing the principle of minimum cross-entropy within a conditional-logical framework. *Artificial Intelligence*, 98:169–208, 1998.

[KI01] G. Kern-Isberner. *Conditionals in nonmonotonic reasoning and belief revision*. Springer, Lecture Notes in Artificial Intelligence LNAI 2087, 2001.

[KI02] G. Kern-Isberner. Handling conditionals adequately in uncertain reasoning and belief revision. *Journal of Applied Non-Classical Logics*, 12(2):215–237, 2002.

[Mül04] C. Müller. Implementing default rules by optimal conditional ranking functions. B.Sc. Thesis, Dept. of Computer Science, FernUniversität in Hagen, Germany, 2004. (in German).

[Par94] J.B. Paris. *The uncertain reasoner's companion – A mathematical perspective*. Cambridge University Press, 1994.

[PV97] J.B. Paris and A. Vencovska. In defence of the maximum entropy inference process. *International Journal of Approximate Reasoning*, 17(1):77–103, 1997.

[Spo88] W. Spohn. Ordinal conditional functions: a dynamic theory of epistemic states. In W.L. Harper and B. Skyrms, editors, *Causation in Decision, Belief Change, and Statistics, II*, pages 105–134. Kluwer Academic Publishers, 1988.

[Wey98] E. Weydert. System JZ - How to build a canonical ranking model of a default knowledge base. In *Proceedings KR'98*. Morgan Kaufmann, 1998.