InfOCF-Lib: A Java Library for OCF-based Conditional Inference

Steven Kutsch

Dept. of Computer Science, FernUniversität in Hagen, Hagen, Germany
steven.kutsch@fernuni-hagen.de

Abstract. Conditionals of the form "If A, then usually B" are often used to define nonmonotonic inference relations. Typically, a finite set of conditionals $\mathcal{R}$, called a knowledge base, is inductively completed to an inference relation containing all the explicit and implicit beliefs of an intelligent agent. One way of representing such a complete epistemic state is by a ranking function, assigning degrees of disbelief to propositional interpretations. Each ranking function defines a nonmonotonic inference relation and each set of ranking functions defines several inference relations by employing different modes of inference. We propose the Java library InfOCF-Lib that is capable of reading a knowledge base from a file, calculating various sets of ranking functions proposed in the literature, and answering queries of the form "Does A entail B in the context of the knowledge base $\mathcal{R}$ using the set of ranking functions M?".

Keywords: conditional logic, conditional knowledge base, ranking function, nonmonotonic inference, Java

1 Introduction

Rules with possible exceptions, i.e. statements of the form "if A, then usually B", play a prominent role in the area of knowledge representation and reasoning. Several semantics have been proposed for defining nonmonotonic inference relations over sets of such rules. Examples are Lewis' system of spheres [17], conditional objects evaluated using Boolean intervals [9], or possibility measures [8, 10].

Here, we will consider Spohn's ranking functions [20] that assign degrees of disbelief to propositional interpretations. C-Representations [13, 14] are a subset of all ranking functions for a knowledge base, which can conveniently be calculated by solving a constraint satisfaction problem dependent on the knowledge base. A first Prolog implementation of this approach was introduced in [4].

In [3], the tool InfOCF was introduced, that allows the user to load knowledge bases, calculate admissible ranking functions (in particular c-representations and system Z [19]), and perform inference using these sets of ranking functions. System P entailment [1] was also implemented. For calculating c-representations, InfOCF interfaced with a Prolog component akin to the one introduced in [4].

While InfOCF supports many tasks when working with conditional knowledge bases, the need for new features or variants of old features arose frequently, as new use cases became relevant in the active research of conditionals and inference relations defined by them. Since it is impossible to foresee every possible use case for a tool such as InfOCF, and because the architecture of InfOCF did not support a practical API, we developed the library InfOCF-Lib, that allows the user to quickly write personal tools for any use case, using the core functionalities of InfOCF. Thus, InfOCF-Lib provides a redesign of InfOCF that is more easily extensible and provides a convenient API. InfOCF-Lib can be accessed online[1] and comes with a detailed user manual.

Other implementations for conditional inference have been proposed. Z-log [18] offers inference based on system Z. The Java library for conditional logic of The Tweety Project[2] offers some limited capabilities for calculating c-representations, i.e. calculation of a single c-representation for a given knowledge base, as well as System Z inference. Since the implementation of rank based inference in The Tweety Project, several new theoretical approaches have been proposed, such as different notions of minimal c-representations [2] and inference using sets of ranking functions [5, 7]. These new approaches have been implemented in InfOCF-Lib.

2 Background

2.1 Conditional Logic and OCFs

Let $\mathcal{L}$ be a propositional language over a finite set Σ of atoms $a_1, \ldots, a_n$. The formulas of $\mathcal{L}$ will be denoted by letters $A, B, C, \ldots$. We write AB for $A \wedge B$ and $\overline{A}$ for $\neg A$. We identify the set of all complete conjunctions over Σ with the set Ω of possible worlds over $\mathcal{L}$. For $\omega \in \Omega$, $\omega \models A$ means that $A \in \mathcal{L}$ holds in ω.

A conditional $(B|A)$ formalises a statement of the form "*if A then usually B*" and establishes a plausible connection between the *antecedent* A and the *consequence* B. It partitions 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 to which the conditional may not be applied to at all. This allows us to associate to $(B|A)$ a *generalised indicator function* $\chi_{(B|A)}$ going back to [11] (where u stands for *unknown* or *indeterminate*):

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

InfOCF-Lib uses Spohn's *ranking functions* (also called ordinal conditional functions, OCFs) [20]. A ranking function is a function $\kappa : \Omega \to \mathbb{N}$ expressing

[1] www.fernuni-hagen.de/wbs/data/InfOCF-Lib.zip
[2] tweetyproject.org

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

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

A ranking function κ *accepts* a conditional $(B|A)$ (denoted by $\kappa \models (B|A)$) if the verification of the conditional is less surprising than its falsification, i.e., if $\kappa(AB) < \kappa(A\overline{B})$. Every ranking function κ gives rise to a nonmonotonic inference relation $\mathrel{|\!\sim^{\kappa}}$ defined as

$$A \mathrel{|\!\sim^{\kappa}} B \quad \text{iff} \quad A \equiv \bot \quad \text{or} \quad \kappa(AB) < \kappa(A\overline{B}) \tag{3}$$

A non-empty finite set of conditionals $\mathcal{R}$ is called a *knowledge base*. An OCF κ accepts $\mathcal{R}$ if κ accepts all conditionals in $\mathcal{R}$, and $\mathcal{R}$ is *consistent* if an OCF accepting $\mathcal{R}$ exists [12]. To test a conditional knowledge base for consistency, an algorithm using the notion of tolerance has been proposed.

Definition 1 (Tolerance [12]). *Let $\mathcal{R} = \{(B_1|A_1), \ldots, (B_n|A_n)\}$ be a knowledge base and $r = (B|A)$ a conditional. The knowledge base $\mathcal{R}$* tolerates *the conditional r if there is an interpretation ω such that*

$$\omega \models AB \wedge \bigwedge_{1 \leqslant i \leqslant n} (\overline{A_i} \vee B_i) \tag{4}$$

The algorithm OrderedPartition (Listing 1.1) calculates the inclusion maximal ordered partition $\mathcal{R}_p = (\mathcal{R}_0, \ldots, \mathcal{R}_k)$ such that for every $0 \leqslant i \leqslant k$, every $r \in \mathcal{R}_i$ is tolerated by $\bigcup_{j=i}^{k} \mathcal{R}_j$. Since OrderedPartition returns NULL if $\mathcal{R}$ is inconsistent, it can be used as a consistency test [12].

2.2 Inference Modes for Sets of Ranking Functions

Typically, there is more then one valid viewpoint when representing a domain that involves uncertainty. We can model reasoning with multiple viewpoints (ranking functions) by performing skeptical, weakly skeptical [2] or credulous inference over a set of ranking functions M.

Definition 2. *Let $\mathcal{R}$ be a knowledge base, M a set of ranking functions accepting $\mathcal{R}$, and A and B be formulas.*

B is a skeptical inference over M from A, denoted by $A \mathrel{|\!\sim^{sk,M}_{\mathcal{R}}} B$, if for all $\kappa \in M$ it holds that $A \mathrel{|\!\sim^{\kappa}} B$.

B is a credulous inference over M from A, denoted by $A \mathrel{|\!\sim^{cr,M}_{\mathcal{R}}} B$, if there is a $\kappa \in M$ such that $A \mathrel{|\!\sim^{\kappa}} B$ holds.

B is a weakly skeptical inference over M from A, denoted by $A \mathrel{|\!\sim^{ws,M}_{\mathcal{R}}} B$, if there is a $\kappa \in M$ such that $A \mathrel{|\!\sim^{\kappa}} B$ holds and there is no $\kappa \in M$ such that $A \mathrel{|\!\sim^{\kappa}} \overline{B}$ holds.

Listing 1.1. Algorithm to test for consistency of $\mathcal{R}$ (cf. [12]).

```
PROCEDURE: OrderedPartition
 INPUT  : Knowledge base R={(B_1|A_1),...,(B_n|A_n)}
 OUTPUT : Ordered partition (R_0,R_1,...,R_k) if R is consistent,
      NULL otherwise

 INT i:=0;
 WHILE(R ≠ ∅) DO
   R_i := { (B|A) ∈ R | R tolerates (B|A) };
   IF(R_i ≠ ∅)
   THEN
     R:=R \ R_i;
     i:=i+1;
   ELSE
     RETURN NULL; // R is inconsistent
 RETURN R_p = (R_0,...,R_k);
```

2.3 C-Representations

In this section we will discuss *c-representations* [13, 14] that assign an individual impact η_i to each conditional $(B_i|A_i)$ and generate the world ranks as a sum of impacts of falsified conditionals:

Definition 3 (c-representation [13, 14]). *A* c-representation *of a knowledge base $\mathcal{R} = \{(B_1|A_1), \ldots, (B_n|A_n)\}$ is an OCF κ constructed from non-negative integer impacts $\eta_i \in \mathbb{N}_0$ assigned to each conditional $(B_i|A_i)$ such that κ accepts $\mathcal{R}$ and is given by:*

$$\kappa(\omega) = \sum_{\substack{1 \leqslant i \leqslant n \\ \omega \models A_i \overline{B}_i}} \eta_i \tag{5}$$

C-representations can conveniently be specified using a constraint satisfaction problem (for detailed explanations, see [13, 14, 5]):

Definition 4 ($CR(\mathcal{R})$ [4]). *Let $\mathcal{R} = \{(B_1|A_1), \ldots, (B_n|A_n)\}$. The constraint satisfaction problem for c-representations of $\mathcal{R}$, denoted by $CR(\mathcal{R})$, on the constraint variables $\eta_1, \ldots, \eta_n$ ranging over $\mathbb{N}$ is given by the conjunction of the constraints, for all $i \in \{1, \ldots, n\}$:*

$$\eta_i > \min_{\omega \models A_i B_i} \sum_{\substack{j \neq i \\ \omega \models A_j \overline{B_j}}} \eta_j - \min_{\omega \models A_i \overline{B_i}} \sum_{\substack{j \neq i \\ \omega \models A_j \overline{B_j}}} \eta_j \tag{6}$$

A solution of $CR(\mathcal{R})$ is a vector $\vec{\eta} = (\eta_1, \ldots, \eta_n)$ of n natural numbers. With $Sol(CR(\mathcal{R}))$ we denote the set of solutions of $CR(\mathcal{R})$. For $\vec{\eta} \in Sol(CR(\mathcal{R}))$ and κ as in Equation (5), κ is the *OCF induced by* $\vec{\eta}$, denoted by $\kappa_{\vec{\eta}}$.

Since there are in general infinitely many solutions of $CR(\mathcal{R})$, three notions of minimality of impact vectors have been proposed [2].

Definition 5 (sum-, cw- and ind-minimality [2]). *Let $\mathcal{R}$ be a knowledge base, and $\vec{\eta}, \vec{\eta} \in Sol(CR(\mathcal{R}))$. Then*

$$\vec{\eta} \preccurlyeq_+ \vec{\eta}' \quad \textit{iff} \quad \sum_{1 \leqslant i \leqslant n} \eta_i \leqslant \sum_{1 \leqslant i \leqslant n} {\eta'}_i \tag{7}$$

$\vec{\eta}$ is sum-minimal *iff $\vec{\eta} \preccurlyeq_+ \vec{\eta}'$ for all $\vec{\eta}' \in Sol(CR(\mathcal{R}))$. We write $\vec{\eta} \prec_+ \vec{\eta}'$ iff $\vec{\eta} \preccurlyeq_+ \vec{\eta}'$ and $\vec{\eta}' \not\preccurlyeq_+ \vec{\eta}$.*

$$\begin{aligned} \vec{\eta} \preccurlyeq_{cw} \vec{\eta}' \quad \textit{iff} \quad & \eta_i \leqslant {\eta'}_i \textit{ for all } i \in \{1, \ldots, n\} \\ & \textit{and } \eta_i < \eta'_i \textit{ for some } i \in \{1, \ldots, n\} \end{aligned} \tag{8}$$

$\vec{\eta}$ is cw-minimal *(or* Pareto-minimal*) iff there is no vector $\vec{\eta}' \in Sol(CR(\mathcal{R}))$ such that $\vec{\eta}' \preccurlyeq_{cw} \vec{\eta}$ and $\vec{\eta} \not\preccurlyeq_{cw} \vec{\eta}'$.*

$$\vec{\eta} \preccurlyeq_O \vec{\eta}' \quad \textit{iff} \quad \kappa_{\vec{\eta}}(\omega) \leqslant \kappa_{\vec{\eta}'}(\omega) \textit{ for all } \omega \in \Omega \tag{9}$$

$\vec{\eta}$ is ind-minimal *iff there is no vector $\vec{\eta}' \in Sol(CR(\mathcal{R}))$ such that $\vec{\eta}' \preccurlyeq_O \vec{\eta}$ and $\vec{\eta} \not\preccurlyeq_O \vec{\eta}'$.*

Since none of the three order relations is total, there are in general multiple sum-, cw-, and ind-minimal c-representations for a given knowledge base. We can therefor perform skeptical, credulous and weakly skeptical inference over four sets of c-representations: all solutions of $CR(\mathcal{R})$, or only sum-, cw-, or ind-minimal solutions.

3 Architecture

In this section we will detail the architecture of `InfOCF-Lib`. Since the implementation of the answering of queries using finte sets of ranking functions is straightforward (loop over ranking functions and check rank of verification and falsification), we will not go into detail about the concrete implementation of these aspects of the library.

The representation of formulas and elements of classical propositional logic is also straightforward and is detailed in the manual that comes with `InfOCF-Lib`. Here we will detail the architecture of the non classical elements of the library.

Figure 1 shows the package `conditionalLogic` containing the two classes `Conditional` and `ConditionalKnowledgeBase`. Conditionals are composed of two propositional formulas. The evaluation of conditionals $(B|A)$ is realised in the method `conditionalIndicatorFunction` that evaluates the conditional in an interpretation ω according to the generalised indicator function $\chi_{(B|A)}(\omega)$ (cf. (1)). Knowledge bases are collections of conditionals together with a name and a signature. Since `InfOCF-Lib` uses Jasper[3] to interface with a SICStus

[3] `sicstus.sics.se/sicstus/docs/4.1.0/html/sicstus/lib_002djasper.html`

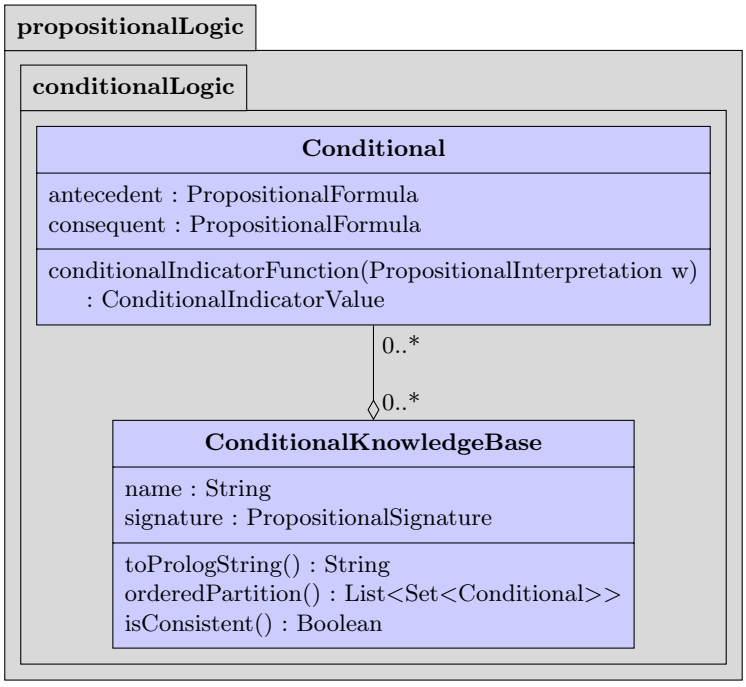

Fig. 1. The package `conditionalLogic` containing the classes `Conditional` and `ConditionalKnowledgeBase`.

Prolog component, knowledge bases need to be translatable to Prolog programs. The method `orderedPartition` calculates the ordered partition according to Algorithm 1.1. The consistency test realised in the method `isConsistent` tries to construct the order partition and returns true or false depending of the existence of the ordered partition.

Figure 2 shows the essential components of `InfOCF-Lib`s representation of ranking functions and sets of ranking functions. As defined in Section 2.1, ranking functions assign integers to every interpretation in a propositional universe (set of all interpretations for a fixed signature). The interpretations for a signature are ordered according to [6]. This allows the correct association with the ranks.

C-Representations (cf. Section 2.3) are ranking functions that assign the rank of interpretations due to falsified conditionals and their impacts. They therefore always need to know the knowledge base $\mathcal{R}$ that they are accepting, together with the impacts, which are obtained from solving $CR(\mathcal{R})$ (see Definition 4). The solving of this constraint problem is implemented in Prolog [3].

In order to represent inference using sets of ranking functions, as defined in Section 2.2, a `RankingFunctionSet` represents a set of ranking functions that accept a common knowledge base. The ranking functions in a `RankingFunctionSet`

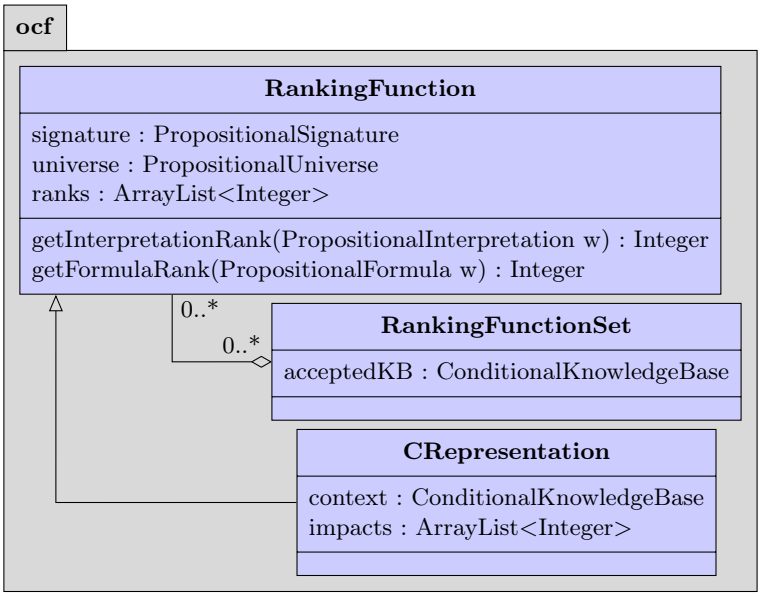

Fig. 2. The package `ocf` containing the classes `RankingFunction`, `CRepresentations` and `RankingFunctionSet`.

also share a common `ModelSetKind`. The Enum `ModelSetKind` identifies the following sets of ranking functions:

- all c-representations
- ind-minimal c-representations
- cw-minimal c-representations
- sum-minimal c-representations
- system Z [19]
- all ranking functions

While the construction of a `RankingFunctionSet` using a `ModelSetKind` representing a set of c-representations always result in a call to the Prolog component for solving $CR(\mathcal{R})$, the `RankingFunctionSet` for `ModelSetKind` representing system Z or all ranking functions are calculated internally. If the constructor of `RankingFunctionSet` is called with the `ModelSetKind` representing all ranking functions, no ranking functions are actually calculated. The resulting `RankingFunctionSet` simply serves as a placeholder object for defining system P inference, which can be characterised as skeptical inference over all ranking functions [16].

Objects of the class `RankedInferenceRelation` (shown in Figure 3) are constructed from

- a set of ranking functions, and

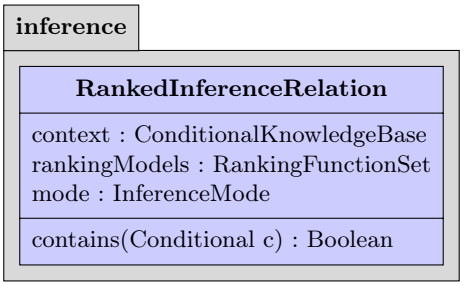

Fig. 3. The class `RankedInferenceRelation` representing inference relations defined over sets of ranking functions.

– a mode of inference (skeptical, weakly skeptical or credulous) (cf. Def. 2).

The context for the inference relation is clear, since the set of ranking functions accepts a common knowledge base. A ranked inference relation represents a set of conditionals, namely all conditionals entailed from the knowledge base by performing inference over the ranking function set with the specified mode of inference. Therefore, the primary method in the class `RankedInferenceRelation` is `contains`, that checks whether the supplied conditional is contained in the inference relation, by performing inference.

If a `RankedInferenceRelation` is constructed with a `RankingFunctionSet` with the `ModelSetKind` for all ranking functions, inference is realised by negating the query conditional r and checking for the consistency of $\mathcal{R} \cup \{\overline{r}\}$ [9].

4 Examples

To give an impression of how `InfOCF-Lib` works, we illustrate the usage with two examples. The first loads a knowledge base from a file, calculates the set of ind-minimal c-representations for the knowledge base and answers a set of queries, also loaded from a file. In the second example, `InfOCF-Lib` is used to load a set of knowledge bases from a file and check every knowledge base for consistency and print the name of every inconsistent knowledge base to the console.

4.1 Loading a Knowledge Base and Answering Queries with Respect to Minimal C-Representations

The knowledge base we want to load is stored in the file `birds.cl` with the following content:

```
signature
   b,p,f
```

```
conditionals
birds001{
   (f | b),  // birds usually fly
   (b | p),  // penguins are birds
   (!f| p)   // penguins don't fly
}
```

In this file, formulas are build up from propositional variables introduced under the keyword signature. The junctors *and* (comma), *or* (semicolon) and *not* (exclamation point) can be used.

To load a file containing a knowledge base, first the content of the file needs to be read into a string using Java's standard functionalities for reading files. We call this string `birdsString`. This string can then be parsed using the functions `InfOCF-Lib` provides.

```
String birdsString = new String(Files.readAllBytes(Paths.get(
    "birds.cl")));
List<ConditionalKnowledgeBase> kbs =
    parseConditionalKnowledgeBase(birdsString);
ConditionalKnowledgeBase birds = kbs.get(0);
```

The method `parseConditionalKnowledgeBase` returns a list of knowledge bases, since .cl-files can contain multiple knowledge bases. We will see the use of these lists in the next example. Here we just select the first and only knowledge base in the file. To answer some queries in the context of this knowledge base, we need to define a `RankedInferenceRelation`. Inference relations are defined using sets of ranking functions that accept a common knowledge base. To generate the set of ind-minimal c-representations accepting the knowledge base `birds`, we simply pass the knowledge base and the parameter specifying ind-minimal c-representations to the constructor of the class `RankingFunctionSet`. The skeptical inference relation over the ind-minimal c-representations accepting the knowledge base `birds` can then be defined.

```
RankingFunctionSet indMinCRepRankingModel =
   new RankingFunctionSet(birds, ModelSetKind.CREP_IND);

RankedInferenceRelation indMinCRepInference =
    new RankedInferenceRelation(indMinCRepRankingModel,
        InferenceMode.SKEPTICAL);
```

In this example, we want to answer the queries in the file `birdsQueries.cl`. The file contains a comma separated list of conditionals:

```
(b | p),
(f | p),
(f | p,b)
```

We can load this list of conditionals in the same way we loaded the knowledge base by reading the content into a string and parsing that string with `InfOCF-Lib`:

```
String queriesString = new String(Files.readAllBytes(Paths.
    get("birdsQueries.cl")));
List<Conditional> queries = parseConditionalQueryList(
    queriesString);
```

We can now iterate over our queries and answer each query using our `RankedInferenceRelation`:

```
for (Conditional query : queries) {
   System.out.print(query.toString() + " : ");
   if (indMinCRepInference.contains(query)) {
      System.out.println("1");
   } else {
      System.out.println("0");
   }
}
```

The code above produces the following output:

```
( b | p ) : 1
( f | p ) : 0
( f | (b,p) ) : 0
```

4.2 Checking a Set of Knowledge Bases for Consistency

Here we again load knowledge bases from a file. The file `birds2.cl` contains multiple knowledge bases:

```
signature
   b,p,f

conditionals
birds001{
   (f | b),  // birds usually fly
   (b | p),  // penguins are birds
   (!f| p)   // penguins don't fly
}

birds002{
   (f | b),  // birds usually fly
   (!f | b)  // birds usually don't fly
}
```

As mentioned above, we will check every knowledge base in this file for consistency, and output the names of the inconsistent knowledge bases.

```
String birdsString = new String(Files.readAllBytes(Paths.get(
    "birds2.cl")));
List<ConditionalKnowledgeBase> kbs =
    parseConditionalKnowledgeBase(birdsString);
```

Iterating over the list of knowledge bases read from the file, we can call the method `isConsistent` for every knowledge base and identify the inconsistent knowledge bases:

```
for (ConditionalKnowledgeBase kb : kbs) {
   if (!kb.isConsistent())
      System.out.println(kb.getName() + " is inconsistent");
}
```

producing the output

```
birds002 is inconsistent
```

5 Conclusions and Future Work

In internal use, `InfOCF-Lib` has proven to be a useful and effective library for working with conditionals and ranking functions. New use cases (e.g. [15]) are quickly realised and experiments using conditional knowledge bases can be designed and conducted easily.

As `InfOCF-Lib` is still in development, many theoretical efficiency benefits are not fully implemented jet. A thorough evaluation of the efficiency of the implementation is part of our future work. We are currently also working on a web based interface for `InfOCF-Lib`, that offers convenient access to the most basic features.

References

1. Adams, E.W.: The Logic of Conditionals: An Application of Probability to Deductive Logic. Synthese Library, Springer Science+Business Media, Dordrecht, NL (1975)
2. Beierle, C., Eichhorn, C., Kern-Isberner, G., Kutsch, S.: Skeptical, weakly skeptical, and credulous inference based on preferred ranking functions. In: ECAI-2016. vol. 285, pp. 1149–1157. IOS Press (2016)
3. Beierle, C., Eichhorn, C., Kutsch, S.: A practical comparison of qualitative inferences with preferred ranking models. KI – Künstliche Intelligenz **31**(1), 41–52 (2017)
4. Beierle, C., Kern-Isberner, G., Södler, K.: 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. LNAI, vol. 7773, pp. 175–192. Springer (2013)
5. Beierle, C., Eichhorn, C., Kern-Isberner, G., Kutsch, S.: Properties of skeptical c-inference for conditional knowledge bases and its realization as a constraint satisfaction problem. Ann. Math. Artif. Intell. **83**(3-4), 247–275 (2018)
6. Beierle, C., Kutsch, S.: Systematic generation of conditional knowledge bases up to renaming and equivalence. In: Calimeri, F., Leone, N., Manna, M. (eds.) Logics in Artificial Intelligence - 16th European Conference, JELIA 2019, Rende, Italy, May 7-11, 2019, Proceedings. LNAI, vol. 11468, pp. 279–286. Springer (2019)

7. Beierle, C., Kutsch, S., Sauerwald, K.: Compilation of conditional knowledge bases for computing c-inference relations. In: Ferrarotti, F., Woltran, S. (eds.) Foundations of Information and Knowledge Systems - 10th International Symposium, FoIKS 2018, Budapest, Hungary, May 14-18, 2018, Proceedings. LNCS, vol. 10833, pp. 34–54. Springer (2018). https://doi.org/10.1007/978-3-319-90050-6_3, https://doi.org/10.1007/978-3-319-90050-6_3
8. Benferhat, S., Dubois, D., Prade, H.: Possibilistic and standard probabilistic semantics of conditional knowledge bases. J. of Logic and Computation **9**(6), 873–895 (1999)
9. Dubois, D., Prade, H.: Conditional objects as nonmonotonic consequence relationships. Special Issue on Conditional Event Algebra, IEEE Transactions on Systems, Man and Cybernetics **24**(12), 1724–1740 (1994)
10. Dubois, D., Prade, H.: Possibility theory and its applications: Where do we stand? In: Kacprzyk, J., Pedrycz, W. (eds.) Springer Handbook of Computational Intelligence, pp. 31–60. Springer, Berlin (2015)
11. de Finetti, B.: La prévision, ses lois logiques et ses sources subjectives. Ann. Inst. H. Poincaré **7**(1), 1–68 (1937), engl. transl. *Theory of Probability*, J. Wiley & Sons, 1974
12. Goldszmidt, M., Pearl, J.: Qualitative probabilities for default reasoning, belief revision, and causal modeling. Artificial Intelligence **84**, 57–112 (1996)
13. Kern-Isberner, G.: Conditionals in nonmonotonic reasoning and belief revision, LNAI, vol. 2087. Springer (2001)
14. Kern-Isberner, G.: A thorough axiomatization of a principle of conditional preservation in belief revision. Ann. of Math. and Artif. Intell. **40(1-2)**, 127–164 (2004)
15. Kutsch, S., Beierle, C.: Computation of Closures of Nonmonotonic Inference Relations Induced by Conditional Knowledge Bases. ECSQARU '19 (2019), (to appear)
16. Lehmann, D.J., Magidor, M.: What does a conditional knowledge base entail? Artificial Intelligence **55**(1), 1–60 (1992)
17. Lewis, D.: Counterfactuals. Harvard University Press, Cambridge, Mass. (1973)
18. Minock, M., Kraus, H.: Z-log: Applying system-z. In: Proceedings of the 8th European Conference on Logics in Artificial Intelligence (JELIA'02). pp. 545 – 548 (2002)
19. Pearl, J.: System Z: A natural ordering of defaults with tractable applications to nonmonotonic reasoning. In: Parikh, R. (ed.) Proceedings of the 3rd conference on Theoretical aspects of reasoning about knowledge (TARK1990). pp. 121–135. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1990)
20. Spohn, W.: The Laws of Belief: Ranking Theory and Its Philosophical Applications. Oxford University Press, Oxford, UK (2012)