Normative Systems Represented as Hybrid Knowledge Bases

Marco Alberti, Ana Sofia Gomes, Ricardo Gonçalves,
João Leite, and Martin Slota

CENTRIA & Departamento de Informática, Universidade Nova de Lisboa, Portugal

Abstract. Normative systems have been advocated as an effective tool to regulate interaction in multi-agent systems.
Logic programming rules intuitively correspond to conditional norms, and their semantics is based on the closed world assumption, which allows default negation, often used in norms. However, there are cases where the closed world assumption is clearly not adequate, and others that require reasoning about unknown individuals, which is not possible in logic programming.
On the other hand, description logics are based on the open world assumption and support reasoning about unknown individuals, but do not support default negation.
In this paper, we demonstrate the need for the aforementioned features (closed and open world assumptions, and reasoning about unknown individuals) in order to model human laws, with examples from the Portuguese Penal Code. We advocate the use of hybrid knowledge bases combining rules and ontologies, which provide the joint expressivity of logic programming and description logics.
We define a normative scenario as the pair of a set of facts and a set of norms, and give it a formal semantics by translation into an MKNF knowledge base.
We describe the implementation of the language, which computes the relevant consequences of given facts and norms, and use it to establish the resulting sentence in a penal scenario.

1 Introduction

In this paper we argue for the need to jointly use the Closed World Assumption based features of Logic Programming Rules, and the Open World Assumption based features of Description Logic based Ontologies, to represent and reason about Norms. We present a solution grounded on Hybrid MKNF Knowledge Bases [18], illustrate its use with an excerpt of the Portuguese Penal Code, and describe an efficient implementation.

Normative systems have long been advocated as an effective tool to regulate interaction in multi-agent systems [24], and the theory and practice of normative multi-agent systems constitutes a young, but very active, research area [6].

Essentially, norms encode desirable behaviours for the population of natural or artificial societies. For example, a (conditional) norm might specify that drivers are expected to stop if so signaled by an authority. In general, they are commonly understood as a specification of what is expected to follow (obligations, goals, contingency plans, advices, actions, ...) from a specific state of affairs.

In practical multi-agent systems, norms are often implemented through electronic institutions which take a formal representation of the normative system and, through automated reasoning, check observable agents' behaviours in order, for instance, to detect norm violation and to apply sanctions.

One key problem to implement such practical normative systems involves the representation of, and reasoning with norms. If, on the one hand, we need a representation language that is expressive enough to represent the norms we wish to encode, on the other hand it must be such that we can reason with it efficiently. In this paper, we will take a closer look at the problem of norm representation and reasoning, trying to combine expressivity and efficiency, which has proved a difficult task in automated reasoning.

Despite the specificities of multi-agent systems, many of their aspects are inspired by human societies, and an intimate parallel between laws in real world legal systems and norms in multi-agent systems can often be drawn.

Ever since the formalisation of the British Nationality Act using Logic Programming by Sergot et al. [23], *non-monotonic* formalisms have been used to deal with many aspects of legal rules and regulations. Important work on this topic also includes the early use of argument-based extended logic programming with defeasible priorities by Prakken and Sartor [20] and the use of defeasible logic by Governatori et al. [12].

The non-monotonic features common to the languages used in these approaches, which implement the Closed-World Assumption, have been shown necessary in the context of reasoning with laws and regulations, for example to represent exceptions.

In this paper, instead of tailoring an artificial multi-agent based scenario to illustrate our points, we will use the Portuguese Penal Code which is filled with examples rich in intrinsic subtleties.

Example 1. The Portuguese Penal Code[1] defines the penalty for murder as follows:

> **Article 131. Murder**
> *Who kills another person shall be punished with imprisonment from eight to sixteen years.*

However, exceptional circumstances for murder increase the duration of the conviction:

[1] Translation by the authors. For the original text, in Portuguese, consult, for example, `http://www.portolegal.com/CPENAL.htm`.

Article 132. Aggravated murder

1. *If death is produced in circumstances which present a special reprehensibility or perversity, the agent is punished with imprisonment of twelve to twenty-five years.*
2. *Is likely to reveal the special perversity or reprehensibility referred to in the preceding paragraph,* among others*, the fact that the agent:*
   - *(...)*
   - *d) employs torture or cruel act to increase the suffering of the victim;*
   - *(...)*
   - *h) performs the fact with at least two other people, or uses particularly dangerous means, or [means] which would result in the crime of common danger;*
   - *(...)*

Accordingly, killing someone is punished with imprisonment from eight to sixteen years, *except* if some additional facts are established, in which case the penalty is aggravated. In other words, unless one of these aggravating facts is proved, *by default* this crime is punished with imprisonment from eight to sixteen years. The relevant part can easily be captured by Logic Programming rules using non-monotonic default negation as follows:

$$\mathsf{AggravatedMurder}(X,Y) \leftarrow \mathsf{KillingBy}(X,Y), \mathsf{Censurable}(X).$$
$$\mathsf{Murder}(X,Y) \leftarrow \mathsf{KillingBy}(X,Y), {\sim}\mathsf{AggravatedMurder}(X,Y).$$

together with the definition of $\mathsf{Censurable}(X)$, which is the predicate representing the "special perversity or reprensehibility" referred to in the law.

The use of non-monotonic rule based languages founded on Logic Programming brings added value, as they also have well-studied computational properties and mature implementations.

However, in legal reasoning, we sometimes need to represent concepts that cannot be handled by the Logic Programming approach. There are cases where the Open World Assumption is needed and, more importantly, others where we need to deal with existential knowledge and unknown individuals.

Example 2. Going back to the previous example, encoding item h) as a condition to establish special perversity or reprehensibility requires that we refer to (at least) two possibly unknown individuals (a witness or a security camera recording could be sufficient to establish that the culprit acted together with two more people, but not their identity). The relevant part could be encoded in Description Logics as follows:

$$\mathsf{Censurable} \sqsupseteq (\geq 3\ \mathsf{PerformedBy.Person})$$

encoding that special censurability of the fact is established if it was committed by at least three people. Such a condition cannot be expressed in the body of a

logic programming rule since it does not permit encoding unknown individuals. Just as it would not be possible to assert that some fact was performed by e.g. five people, but whose identities, besides that of the accused, are unknown.

Description Logics [2] based ontology languages are based on the Open World Assumption and allow for reasoning with unknown individuals. Furthermore, they are quite appropriate for taxonomic representations of facts and concepts, and have been extensively used in legal reasoning [10]. However, they are monotonic and therefore lack the aforementioned important ability to model defeasible knowledge.

For all of these reasons, the representation and reasoning about normative systems, and in particular those inspired by human legal systems, seems to demand an approach that combines the best of the two families of formalisms, rules and ontologies, and exhibits, at least, the following characteristics:

- have a formal rigorous semantics so that agents and institutions can both reason about the norms to determine their actions and sanctions;
- support both the Open and Closed World Assumptions, and the ability to represent and reason with unknown individuals;
- be equipped with efficient operational semantics to be usable in practical multi-agent systems.

With these requirements in mind, in this paper we propose a language where facts are represented as a description logic ABox, and norms as a combination of description logic TBox and logic programming rules. We will then root our language on Hybrid MKNF [18], a language that tightly integrates rules and ontologies, and focus on its well founded version [1, 14] for which we can exploit an efficient implementation [11].

The remainder of the paper is structured as follows. In Sect. 2, we present the language and the running example from the Portuguese penal code. In Sect. 3, we give the language a formal semantics by mapping it to Hybrid MKNF. In Sect. 4, we introduce the implementation of the reasoner for our language, and show how it correctly handles the running example. Conclusions and comments on future research follow.

2 Framework

In this section, we introduce the model of electronic institution that we envisage by analogy with the Portuguese judicial system; the institution's modus operandi motivates the choice of a language for a normative system.

In Portugal, the first phase of a penal trial is a discussion aimed at establishing the facts, where each of the parties (prosecution and defense) provides relevant evidence.

At the end of the debate, a list is made of all the facts that have been established in the debate. For example, it may have been established that (*i*) Bob attempted murder on Mary by means of a knife, and (*ii*) Alice defended Mary's life by killing Bob with a handgun.

Then, the judge applies the relevant laws to the facts and pronounces a sentence.

This procedure implies that the sentence depend on (*i*) facts established in court; and (*ii*) relevant laws.

A similar model is in fact applied to normative multi-agent systems, where an electronic institution automatically determines possible sanctions depending on the agents' behaviour (possibly detected by a runtime monitoring system) and the norms.

In order to enable the electronic institution to perform this task by means of automated reasoning, the language used to express the facts and the norms must have a formal semantics.

2.1 Language

We record both the facts about a particular legal case as well as the relevant norms for applying the law in a *judicial knowledge base*. The formalisation of this notion must build upon a formalism that, on the one hand, is capable of reasoning with unknown individuals, and, on the other hand, allows for expressing exceptions in a natural way. At the same time, the formalism should be amenable to an efficient implementation. The natural candidates for such formalisms are the integrative formalisms for Description Logics and non-monotonic rules. We proceed by briefly introducing the syntax of both these formalisms.

Throughout the paper we use a function-free first-order signature. By a *first-order atom* we mean a formula $P(t_1, t_2, \ldots, t_n)$ where P is a predicate symbol of arity n and $t_1, t_2, \ldots, t_n$ are first-order terms.

Description Logics. Description Logics (DLs) [2] are (usually) decidable fragments of first-order logic that are frequently used for knowledge representation and reasoning in applications. Throughout the paper we assume that some Description Logic is used to describe an ontology, i.e. it is used to specify a shared conceptualisation of a domain of interest. Basic building blocks of such a specification are *constants*, representing objects (or individuals), *concepts*, representing groups of objects, and *roles*, representing binary relations between objects and properties of objects. Typically, an ontology is composed of two distinguishable parts: a TBox specifying the required terminology, i.e. concept and role definitions, and an ABox with assertions about constants.

Most Description Logics can be equivalently translated into function-free first-order logic, with constants represented by constant symbols, atomic concepts represented by unary predicates and atomic roles represented by binary predicates. We assume that for any DL axiom ϕ, $\zeta(\phi)$ denotes such a translation of ϕ.

Logic Programs. We consider ground logic programs for specifying nonmonotonic domain knowledge. The basic syntactic blocks of such programs are ground atoms. A *default literal* is a ground atom preceded by $\sim$. A *literal* is either a ground atom or a default literal. A *rule* r is an expression of the form

$$p \leftarrow p_1, p_2, \ldots, p_m, \sim q_1, \sim q_2, \ldots, \sim q_n.$$

where m, n are natural numbers and $p, p_1, p_2, \ldots, p_m, q_1, q_2, \ldots, q_n$ are first-order atoms. The atom p is called the *head of* r and is denoted by $H(r)$; the set $\{ p_1, p_2, \ldots, p_m, \sim q_1, \sim q_2, \ldots, \sim q_n \}$ is called the *body of* r and is denoted by $B(r)$. A rule r is a *fact* if its body is empty. A *(normal) logic program* $\mathcal{P}$ is a finite set of rules.

Judicial Knowledge Base. As illustrated in Examples 1 and 2, a combination of TBox axioms and rules is sufficient to faithfully represent significant parts of real-world penal code. Also, ABox assertions are a natural candidate for describing the facts established in court. We henceforth define a judicial knowledge base as follows:

Definition 1. *A* judicial knowledge base *is a pair* $\langle \mathcal{F}, \mathcal{N} \rangle$ *where* $\mathcal{F}$ *is a set of ABox axioms and* $\mathcal{N}$ *is a set of TBox axioms and rules.*

2.2 Example

In this section, we demonstrate the expressivity of our language by encoding one example from real-world law, which would be problematic to express in either logic programming or description logic formalisms.

Together with the definitions of murder and aggravated murder reported in the introduction, consider the circumstances that exclude the illegality of an act listed in the Portuguese Penal Code.

> **Article 31. Precluding wrongfulness**
>
> 1. *The act is not punishable when its wrongfulness is precluded by law considered in its entirety.*
> 2. *In particular, it (the act) is not unlawful if committed:*
>   a) *In legitimate defense;*
>   b) *In the exercise of a right;*
>   c) *In fulfilling a duty imposed by law or lawful order of the authority, or*
>   d) *With the consent of the holder of the harmed legal interest.*

Among such circumstances, legitimate defense is defined as follows.

> **Article 32. Legitimate defense**
> *Legitimate defense is an act executed as necessary means to repel an actual and illicit aggression to legally protected interest of the executor or of a third party.*

Suppose that the following was established in court by debate:

- John committed murder together with three more people;
- Mary killed someone who was pointing a gun at her;
- an unidentified person committed murder and torture.

A formalisation using a judicial knowledge base $\mathcal{K} = \langle \mathcal{F}, \mathcal{N} \rangle$ is in Figs. 1 and 2.

$$
\begin{aligned}
\mathsf{Censurable} &\sqsupseteq (\geq 3\ \mathsf{PerformedBy.Person}) \\
\mathsf{Censurable} &\sqsupseteq \mathsf{Torture} \\
\mathsf{AggravatedMurder}(X,Y) &\leftarrow \mathsf{KillingBy}(X,Y), \mathsf{Censurable}(X). \\
\mathsf{Murder}(X,Y) &\leftarrow \mathsf{KillingBy}(X,Y), {\sim}\mathsf{AggravatedMurder}(X,Y). \\
\mathsf{KillingBy}(X,Y) &\leftarrow \mathsf{Killing}(X), \mathsf{CrimeBy}(X,Y). \\
\mathsf{CrimeBy}(X,Y) &\leftarrow \mathsf{Illegal}(X), \mathsf{Guilt}(X,Y). \\
\mathsf{Illegal}(X) &\leftarrow \mathsf{Unlawful}(X), {\sim}\mathsf{ExceptionIllegal}(X). \\
\mathsf{Unlawful}(X) &\leftarrow \mathsf{Killing}(X). \\
\mathsf{ExceptionIllegal}(X) &\leftarrow \mathsf{JudicialOrder}(X). \\
\mathsf{ExceptionIllegal}(X) &\leftarrow \mathsf{ExerciseOfARight}(X). \\
\mathsf{ExceptionIllegal}(X) &\leftarrow \mathsf{WithConsentOfVictim}(X). \\
\mathsf{ExceptionIllegal}(X) &\leftarrow \mathsf{LegitimateDefense}(X). \\
\mathsf{LegitimateDefense}(X) &\leftarrow \mathsf{RelevantAggression}(Z), \mathsf{EffectiveReaction}(X,Z), \\
&\qquad {\sim}\mathsf{RequiredDifferentBehaviour}(X,Z). \\
\mathsf{Guilt}(X,Y) &\leftarrow \mathsf{Illegal}(X), \mathsf{PerformedBy}(X,Y), {\sim}\mathsf{ExceptionGuilt}(X,Y). \\
\mathsf{ExceptionGuilt}(X,Y) &\leftarrow \mathsf{RelevantThreat}(Z), \mathsf{EffectiveReaction}(X,Z), \\
&\qquad {\sim}\mathsf{RequiredDifferentBehaviour}(Y,X,Z). \\
\mathsf{JailSentence}(Y,8,16) &\leftarrow \mathsf{Murder}(X,Y), {\sim}\mathsf{AggravatedMurder}(X,Y). \\
\mathsf{JailSentence}(Y,12,25) &\leftarrow \mathsf{AggravatedMurder}(X,Y).
\end{aligned}
$$

Fig. 1. Set of relevant norms $\mathcal{N}$

$a : \mathsf{Act}$	$a : \mathsf{Killing}$	$a : (\geq 4\ \mathsf{PerformedBy.Person})$
$john : \mathsf{Person}$	$\langle a, john\rangle : \mathsf{PerformedBy}$	
$b : \mathsf{Act}$	$b : \mathsf{Killing}$	
$mary : \mathsf{Person}$	$\langle b, mary\rangle : \mathsf{PerformedBy}$	
$gun : \mathsf{RelevantThreat}$	$\langle b, gun\rangle : \mathsf{EffectiveReactionTo}$	
$c : \mathsf{Act}$	$c : \mathsf{Killing}$	$c : \mathsf{Torture}$

Fig. 2. Set of facts established in court $\mathcal{F}$

3 Formal Semantics

In order for judicial knowledge bases to be used in determining sanctions by means of automated reasoning in a rigorous and verifiable fashion, they need a

formal semantics. This semantics is established here by translating them into the logic of Minimal Knowledge and Negation as Failure (MKNF) [16, 18]. In the following, we first briefly introduce the syntax and semantics of MKNF, adopting the well-founded MKNF model semantics introduced in [15] for which top-down querying procedures have been introduced and an implementation with support for the Description Logic $\mathcal{ALCQ}$ is available [11]. In Sect. 4, we use this implementation to represent and query the judicial knowledge base in Figs. 1 and 2.

3.1 Hybrid MKNF Knowledge Bases

Syntactically, MKNF is an extension of function-free first-order logic with two modal operators: **K** and **not**. A *first-order atom* $P(t_1, t_2, \ldots, t_n)$ is an MKNF formula; if ϕ is an MKNF formula, then $\neg\phi$, $\exists x : \phi$, $\mathbf{K}\,\phi$, and $\mathbf{not}\,\phi$ are MKNF formulas, and so are $\phi_1 \wedge \phi_2$, $\phi_1 \vee \phi_2$, $\phi_1 \supset \phi_2$, and $\phi_1 \equiv \phi_2$, if ϕ_1, ϕ_2 are MKNF formulas. An MKNF formula ϕ is a *sentence* if it has no free variables. By $\phi[t/x]$ we denote the formula obtained by simultaneously replacing in ϕ all free occurrences of variable x by term t.

Hybrid MKNF knowledge bases, as presented in [15], are sets of MKNF formulas restricted to a certain form. They consist of two components: a description logic knowledge base and a logic program.[2] Formally, a *hybrid knowledge base* is a pair $\langle \mathcal{O}, \mathcal{P} \rangle$ where $\mathcal{O}$ is a DL ontology and $\mathcal{P}$ is a logic program.

For interpreting hybrid MKNF knowledge bases in terms of the logic of MKNF, a transformation π is defined that translates both ontology axioms and rules into MKNF sentences. For any ground atom p, set of literals B, rule r with the vector of free variables $\boldsymbol{x}$, and hybrid knowledge base $\mathcal{K} = \langle \mathcal{O}, \mathcal{P} \rangle$, we define:

- $\pi(p) = \mathbf{K}\,p$,
- $\pi(\sim p) = \mathbf{not}\,p$,
- $\pi(B) = \{\, \pi(L) \mid L \in B \,\}$,
- $\pi(r) = (\forall \boldsymbol{x} : \bigwedge \pi(B(r)) \supset \pi(H(r)))$,
- $\pi(\mathcal{O}) = \bigwedge \{\, \zeta(\phi) \mid \phi \in \mathcal{O} \,\}$,
- $\pi(\mathcal{P}) = \bigwedge \{\, \pi(r) \mid r \in \mathcal{P} \,\}$,
- $\pi(\mathcal{K}) = \mathbf{K}\,\pi(\mathcal{O}) \wedge \pi(\mathcal{P})$.

Hybrid MKNF knowledge bases are in general undecidable, unless they are restricted in some way. The reason for that is that rules can be applied to all the objects in an infinite domain. The basic idea to make reasoning with hybrid MKNF knowledge bases decidable is to apply rules only to the individuals that appear in the knowledge base. This restriction is achieved by DL-safety [19]. Given a hybrid knowledge base $\langle \mathcal{O}, \mathcal{P} \rangle$, a first-order atom $P(t_1, t_2, \ldots, t_n)$ such that P occurs in $\mathcal{O}$ is called a *DL-atom*; all other atoms are called *non-DL-atoms*. A rule r is *DL-safe* if every variable in r occurs in at least one non-DL-atom p occurring positively in the body of r.

[2] In difference to [15], in this paper we consider only non-disjunctive rules.

3.2 Well-Founded MKNF Model

Following [18, 15], we only consider Herbrand interpretations in our semantics and adopt the *standard names assumption*, so apart from the constants used in formulae, we assume our signature to contain a countably infinite supply of constants. The Herbrand Universe of such a signature is denoted by Δ. A *three-valued MKNF structure* $\langle I, \mathcal{M}, \mathcal{N} \rangle$ consists of a first-order interpretation I and two pairs $\mathcal{M} = \langle M, M_1 \rangle$ and $\mathcal{N} = \langle N, N_1 \rangle$ of sets of first-order interpretations where $M_1 \subseteq M$ and $N_1 \subseteq N$. Each three-valued MKNF structure assigns one of the truth values $\mathbf{t}$, $\mathbf{u}$ and $\mathbf{f}$ to all MKNF sentences. We use an ordering $\mathbf{f} < \mathbf{u} < \mathbf{t}$ of these truth values and operators max and min that, respectively, choose the greatest and least element with respect to this ordering. The truth value of a ground atom $P(t_1, t_2, \ldots, t_n)$ and of an MKNF sentence ϕ in a three-valued MKNF structure $\langle I, \mathcal{M}, \mathcal{N} \rangle$ is defined as follows:

- $\langle I, \mathcal{M}, \mathcal{N} \rangle(P(t_1, t_2, \ldots, t_n)) = \begin{cases} \mathbf{t} & \text{iff } \langle t_1^I, t_2^I, \ldots, t_n^I \rangle \in P^I \\ \mathbf{f} & \text{iff } \langle t_1^I, t_2^I, \ldots, t_n^I \rangle \notin P^I \end{cases}$

- $\langle I, \mathcal{M}, \mathcal{N} \rangle(\neg\phi) = \begin{cases} \mathbf{t} & \text{iff } (I, \mathcal{M}, \mathcal{N})(\phi) = \mathbf{f} \\ \mathbf{u} & \text{iff } (I, \mathcal{M}, \mathcal{N})(\phi) = \mathbf{u} \\ \mathbf{f} & \text{iff } (I, \mathcal{M}, \mathcal{N})(\phi) = \mathbf{t} \end{cases}$

- $\langle I, \mathcal{M}, \mathcal{N} \rangle(\phi_1 \wedge \phi_2) = \min \{ \langle I, \mathcal{M}, \mathcal{N} \rangle(\phi_1), \langle I, \mathcal{M}, \mathcal{N} \rangle(\phi_2) \}$

- $\langle I, \mathcal{M}, \mathcal{N} \rangle(\phi_1 \supset \phi_2) = \begin{cases} \mathbf{t} & \text{iff } \langle I, \mathcal{M}, \mathcal{N} \rangle(\phi_2) \geq \langle I, \mathcal{M}, \mathcal{N} \rangle(\phi_1) \\ \mathbf{f} & \text{otherwise} \end{cases}$

- $\langle I, \mathcal{M}, \mathcal{N} \rangle(\exists x : \phi) = \max \{ \langle I, \mathcal{M}, \mathcal{N} \rangle(\phi[\alpha/x]) \mid \alpha \in \Delta \}$

- $\langle I, \mathcal{M}, \mathcal{N} \rangle(\mathbf{K}\,\phi) = \begin{cases} \mathbf{t} & \text{iff } \langle J, \langle M, M_1 \rangle, \mathcal{N} \rangle(\phi) = \mathbf{t} \text{ for all } J \in M \\ \mathbf{f} & \text{iff } \langle J, \langle M, M_1 \rangle, \mathcal{N} \rangle(\phi) = \mathbf{f} \text{ for some } J \in M_1 \\ \mathbf{u} & \text{otherwise} \end{cases}$

- $\langle I, \mathcal{M}, \mathcal{N} \rangle(\mathbf{not}\,\phi) = \begin{cases} \mathbf{t} & \text{iff } \langle J, \mathcal{M}, \langle N, N_1 \rangle \rangle(\phi) = \mathbf{f} \text{ for some } J \in N_1 \\ \mathbf{f} & \text{iff } \langle J, \mathcal{M}, \langle N, N_1 \rangle \rangle(\phi) = \mathbf{t} \text{ for all } J \in N \\ \mathbf{u} & \text{otherwise} \end{cases}$

An *MKNF interpretation* is a non-empty set of first-order interpretations. An *MKNF interpretation pair* $\langle M, N \rangle$ consists of two MKNF interpretations M, N with $\emptyset \subsetneq N \subseteq M$. An MKNF interpretation pair *satisfies* an MKNF sentence ϕ, written $\langle M, N \rangle \models \phi$, if and only if $\langle I, \langle M, N \rangle, \langle M, N \rangle \rangle(\phi) = \mathbf{t}$ for every $I \in M$. If there exists an MKNF interpretation pair satisfying ϕ, then ϕ is *consistent*. An MKNF interpretation pair $\langle M, N \rangle$ is a *three-valued MKNF model* for a given MKNF sentence ϕ if

1. $\langle M, N\rangle$ satisfies ϕ and
2. for each MKNF interpretation pair $\langle M', N'\rangle$ with $M \subseteq M'$ and $N \subseteq N'$, where at least one of the inclusions is proper and $M' = N'$ if $M = N$, there is some $I' \in M'$ such that $\langle I', \langle M', N'\rangle, \langle M, N\rangle\rangle(\phi) \neq \mathbf{t}$.

For any MKNF interpretation pairs $\langle M_1, N_1\rangle$ and $\langle M_2, N_2\rangle$ we define the knowledge ordering as: $\langle M_1, N_1\rangle \succeq_k \langle M_2, N_2\rangle$ iff $M_1 \subseteq M_2$ and $N_1 \supseteq N_2$. Such an order is of particular interest for comparing models. In logic programming the least model w.r.t. derivable knowledge among all three-valued models for a given program is the well-founded model. Here, we introduce a similar notion referring to the minimal three-valued MKNF models, i.e. the ones among all three-valued MKNF models that leave as much as possible undefined.

Definition 2 (Well-Founded MKNF Model). *Let ϕ be an MKNF sentence and $\langle M, N\rangle$ a three-valued MKNF model of ϕ such that $\langle M_1, N_1\rangle \succeq_k \langle M, N\rangle$ for all three-valued MKNF models $\langle M_1, N_1\rangle$ of ϕ. Then $\langle M, N\rangle$ is a* well-founded MKNF model *of ϕ.*

Given a hybrid knowledge base $\mathcal{K}$, the well-founded model of $\mathcal{K}$ *is the well-founded model of $\pi(\mathcal{K})$.*

The following Theorem pinpoints the fact that every consistent DL-safe hybrid MKNF knowledge base has a unique well-founded model.

Theorem 1 (Theorem 1 in [15]). *If $\mathcal{K}$ is a consistent DL-safe hybrid MKNF knowledge base, then a well-founded MKNF model of $\mathcal{K}$ exists, and it is unique.*

As shown in [15], the well-founded MKNF semantics also generalises the well-founded semantics for logic programs [9] – for every logic program $\mathcal{P}$, the well-founded model of $\mathcal{P}$ directly corresponds to well-founded MKNF model of $\pi(\mathcal{P})$.

Finally, the well-founded model of a judicial knowledge base is determined by the hybrid knowledge base to which it directly corresponds.

Definition 3 (Well-Founded Model of a Judicial Knowledge Base). *Let $\mathcal{K} = \langle \mathcal{F}, \mathcal{N}\rangle$ be a judicial knowledge base, $\mathcal{T}$ a TBox and $\mathcal{P}$ a logic program such that $\mathcal{N} = \mathcal{T} \cup \mathcal{P}$. The* hybrid knowledge base associated to $\mathcal{K}$ *is $\mathcal{K}' = \langle \mathcal{F} \cup \mathcal{T}, \mathcal{P}\rangle$.*

The well-founded model of $\mathcal{K}$ *is the well-founded model of $\mathcal{K}'$.*

4 Implementation

As defined in Section 3, the MKNF Semantics is parametric on a decidable description logic in which the ontology is written. As shown in [14], the choice of this description logic determines the usability of the system, as the complexity of reasoning in the well-founded MKNF semantics is in the same class as the decidable description logic; a complexity result that is extended to a query-driven approach in [1].

CDF-Rules[3] [11] is an implementation of the well-founded MKNF semantics, on XSB system, that fixes the description logic component to CDF [27] ontologies which, in its Type-1 version, fully supports the $\mathcal{ALCQ}$ description logic. Since reasoning in $\mathcal{ALCQ}$ is performed in `EXPtime`, *CDF-Rules* also achieves `EXPtime` complexity.

One particular advantage of *CDF-Rules* is that it is a query-driven system. In fact, the definition of the well-founded MKNF semantics presented in Section 3 constructs the complete well-founded model for a given hybrid knowledge base, based on a bottom-up computation. However, in practical cases, this is not what is intended. Particularly, in the context of judicial systems, what one usually wants is to know whether or not a particular law is applicable to a particular individual, or what is the applicable penalty. Deriving all the consequences of the knowledge base to answer a query about an individual, or about a particular crime, would be in most cases impractical.

CDF-Rules provides a practical framework for query evaluation in well-founded MKNF knowledge bases, combining rules and ontologies, which is sound and complete w.r.t. the well-founded semantics. Moreover, since the example presented in Section 2.2 can be expressed using $\mathcal{ALCQ}$ theories, *CDF-Rules* is suitable for our intents.

4.1 *CDF-Rules*

To evaluate queries, *CDF-Rules* makes use of XSB's SLG Resolution [7], together with tableaux mechanisms supported by CDF theorem prover to check entailment in the ontology. Accordingly, to decide the truth value of a formula defined in the rules component, *CDF-Rules* employs SLG Resolution which, in a nutshell, is a tabling method of resolution that is able to answer queries under the well-founded semantics with polynomial-time complexity.

However, to decide the entailment of a formula ϕ w.r.t. an ontology $\mathcal{O}$ a tableau algorithm tries to construct a common *model* for $\neg\phi$ and $\mathcal{O}$, sometimes called a *completion graph* (cf. e.g. [22]). If such a model can not be constructed then $\mathcal{O}$ entails ϕ; otherwise $\mathcal{O}$ does not entail ϕ. Similar to other description logic provers, the CDF theorem prover attempts to traverse as little of an ontology as possible when proving ϕ. As a result, when the prover is invoked on an atom p, the prover attempts to build a model for the underlying individual(s) to which p refers, and explores additional individuals only as necessary.

Given the interdependence between the rules and the ontology in MKNF knowledge bases, the prover must consider the knowledge inferred by the rules in the program for the entailment proof, as a DL-atom can be derived by rules, which in turn may rely on other DL-atoms proven by the ontology. Thus, in order do compute a query Q, *CDF-Rules* constructs a fixed point that iteratively computes a (sub-)model for the objects that appear in Q, deriving at each iteration new information about their roles and classes, along with information about

[3] The implementation is available from the XSB CVS repository as part of the CDF package in the subdirectory `packages/altCDF/mknf`.

other individuals related to them, either in the ontology (via CDF's tableau algorithm) or in the rules (via SLG procedures). Since this (sub-)model is only constructed for objects that are *relevant* to the query, *CDF-Rules* significantly reduces the amount of computation required when compared to the Definition of well-founded MKNF model presented in Section 3.2.

As stated in Section 3, the definition of MKNF imposes the restriction of DL-safe knowledge bases to obtain decidability. The idea of this concept is basically to constrain the use of rules to individuals actually appearing in the knowledge base under consideration. To implement such concept, *CDF-Rules* has two special predicates *definedClass/2* and *definedRole/3* that define the *domain* of the rules, i.e. the set of individuals that are applicable to a given rule. These predicates can be defined explicitly by the compiler or the programmer, but they can also be inferred if all rules are DL-safe.

Next, we provide details on how to implement the example from Section 2.2 in *CDF-Rules*. For more information on the implementation of *CDF-Rules* the interested reader is referred to [11].

4.2 Implementing Judicial Knowledge Bases

Intuitively, to implement a judicial knowledge base (and particularly, the example presented in Section 2.2), one needs to define two different components in *CDF-Rules*: one component defining the logic-programming rules; and one component defining the intensional (TBox) and extensional (ABox) portions of the ontology.

Rules in *CDF-Rules* are stated in a Prolog-like style but with the inclusion of two additional special predicates *known/1* and *dlnot/1* that account for the modalities **K** and **not** of MKNF[4]. For instance, the rules for defining murder and aggravatedMurder presented in Section 2.2 can be represented as illustrated in Figure 3.

```
aggrMurder(X,Y) :- known(killingBy(X, Y)), known(cens(X)).
murder(X,Y)     :- known(killingBy(X, Y)),
                                        dlnot(aggrMurder(X,Y)).
```

Fig. 3. *CDF-Rules* implementation of murder and aggravatedMurder properties

On the other hand, the ontology component in *CDF-Rules* is defined under the standard CDF syntax. Instances of classes, roles and objects in CDF syntax are defined by using the constructs *cid/2*, *rid/2*, *oid/2*, respectively. These

[4] As defined in [11], we can relax the assumption of DL-Safe rules if we assume that the predicates *definedClass/2* and *definedRole/3* are defined for each rule.

constructs are then used in reserved predicates that are interpreted as ontology definitions.

Furthermore, to express more complex statements to define classes as negation or cardinality, CDF-syntax supports a special kind of identifier: *a virtual identifier*, denoted by the functor *vid/1*. This identifier is then used within the special fact *necessCond/2* that has the goal to state necessary conditions.

As illustration, in order to state that the class Torture is included in the class Censurable (i.e., Censurable $\sqsupseteq$ Torture) we use the predicate *isa/2* that defines state inclusion (cf. Figure 4).

```
isa_ext(cid(torture,mknf),cid(cens,mknf)).
```

Fig. 4. CDF syntax for Censurable $\sqsupseteq$ Torture

Moreover, to state the expression Censurable $\sqsupseteq$ ($\geq$ 3 PerformedBy.Person) we need a little tweak. Particularly, since CDF syntax only allows one to state inclusion (via *isa/2*) between identifiers of classes or objects, to express such statement we need to transform the expression ($\geq$ 3 PerformedBy.Person) also into a class identifier. With this goal, we first need to define an auxiliary class Aux such that Aux $\doteq$ ($\geq$ 3 PerformedBy.Person). Since we are defining cardinality of a class, to express such condition we need the special predicate *necessCond/2*. Afterwards, all one needs to do is to state that the auxiliary class definition Aux is a subclass of Censurable. This is materialised in Figure 5.

```
necessCond_ext(cid(aux,mknf),
     vid(atLeast(3,rid(performedBy,mknf),cid(person,mknf)))).
isa_ext(cid(aux,mknf),cid(cens,mknf)).
```

Fig. 5. CDF syntax for Censurable $\sqsupseteq$ ($\geq$ 3 PerformedBy.Person)

Furthermore, to define simple minimal and maximal cardinality of objects in CDF syntax, one can use the predicates *minAttr/4* and *maxAttr/4*, respectively. For instance, the predicate *minAttr/4* can be used to encode the ABox $a : (\geq$ 4 PerformedBy.Person). This is illustrated in Figure 6.

As expected, such implementation is able to answer queries about the individuals *john* and *mary* and about the facts a, b and c. Particularly, one is able to ask questions to the system, in a Prolog-like style, and conclude that *john*

```
minAttr_ext(oid(f,mknf),rid(performedBy,mknf),
                                              cid(person,mknf),4).
```

Fig. 6. CDF syntax for $a : (\geq 4$ PerformedBy.Person)

should be sentenced to 12–25 years in jail, and that *mary* is not guilty of crime b.[5]

5 Conclusions and Future Work

In this paper, we considered human-inspired normative systems and, in particular, those to be employed to regulate interaction in multi-agent systems. We advocated the need for an integration of rules and ontologies (already recognized in other areas of knowledge representation and reasoning) by showing how both are required to express an excerpt of a real-world Penal Code. We presented a language to express established facts (as a description logic ABox) and laws (as a hybrid of logic programming rules and description logic TBox). We provided the language with a formal semantics by means of a mapping to Hybrid MKNF, and exploited an efficient implementation to draw correct conclusions in the motivating example.

In our working example, we have focused on the outcome of applying the relevant laws to a given set of facts established in court. Therefore, the trial discussion phase, where each of the parties (prosecution and defense) provides relevant evidence to establish the facts, is outside of the scope of our framework. A neat approach to deal with the discussion phase is the argumentation framework of Sartor and Prakken [20].

Several other important aspects of rule-based approaches to normative systems and, in particular, to legal reasoning are outside the scope of the present work. Many of them have been acknowledged in the field of artificial intelligence and law, where there is now much agreement about the structure and properties of rules representing norms. These aspects include, for example, updates of norms, jurisdiction, authority, temporal properties, conflicting rules, exclusionary rules, legal interpretation and contrary-to-duty obligations. Notably, the works of Sartor and Prakken [20, 21] using an Argumentation framework, Governatori et al [12, 4] using a Defeasible Logic framework, Makinson and Van der Torre et al [17, 5] using Input-Output Logic, are examples of works that fruitfully tackled some of these problems.

Boella et al. [3], among others, formally support the distinction between *constitutive norms*, which define concepts, and *regulative norms*, which describe desirable behaviour. While this distinction is conceptually clear and most probably

[5] The full encoding of the example presented in Figs. 1 and 2 in *CDF-Rules* is available in `http://centria.di.fct.unl.pt/~jleite/climaXIIexample.zip`

useful for designing artificial normative systems, in this paper we did not follow this approach, because we preferred to remain faithful to the context of our motivating example, the Portuguese Penal Code, where such a formal distinction is not present. However, splitting the normative part of the judicial knowledge base into constitutive and regulatory portions would present to technical problems, and would allow to apply MKNF hybrid knowledge base evolution to support updates of constitutive norms (as done in [3]), regulative norms, or both.

In [8] we can find an application of first-order logic to the problem of specifying and reasoning about information flow and privacy polices. The logic has an expressive language, which allows the support for some features relevant in the area of information flow and privacy polices, and has been used to formalise large fragments of real world privacy laws, such as the Health Insurance Portability and Accountability Act (HIPAA) and the Gramm-Leach-Bliley Act (GLBA). The main difference w.r.t. our proposal is that they use classical implication to model policy rules, whereas we use logic programming implication to model norms. Moreover, their approach does not have a feature which is fundamental in the area of normative systems which is the ability to represent defeasible knowledge, allowed in our proposal by means of the use of default negation in rules.

With respect to future work, one of our main goals is to investigate the important problem of dealing with the evolution of normative systems, in particular those with hybrid representations such as the one described in this paper. The problems associated with knowledge evolution have been extensively studied, over the years, in different research communities, namely in the context of Classical Logic, and in the context of Logic Programming. They proved to be extremely difficult to solve, and existing solutions, even within each community, are still subject of active debate as they do not seem adequate in all kinds of situations in which their application is desirable. Then, when one looks at the solutions developed by the two communities, one immediately realises that they are based on entirely different intuitions and principles which makes them apparently irreconcilable. Two recent proposals [25, 26] towards the development of update operators for Hybrid knowledge Bases seem promising and can perhaps serve as starting points to tackle the problem of dealing with the evolution of normative systems represented as Hybrid Theories.

We also intend to investigate the integration of our reasoner as a normative engine in existing multi-agent platforms, as a runtime compliance verification tool. In particular, we envisage the hybrid TBox and rules part of the judicial knowledge base to be defined by the system designer, and the ABox part to be provided by a monitoring system; the reasoner's task would be to draw conclusions about, for instance, applicable sanctions. Another long-term target of our research is a-priori compliance verification, given a system specification and a normative system. A possible approach would be similar to those proposed in the business process domain by Governatori and Rotolo [13]: compute logical consequences of the system formal model and the norms to which the system will be subject.

Acknowledgments

The authors would like to thank Ana Leite for all her advice on the Portuguese Penal Code and Matthias Knorr for discussions on the well-founded semantics for Hybrid MKNF. Ana Sofia Gomes was partially supported by FCT Grant SFRH/BD/64038/2009. Ricardo Gonçalves was partially supported by FCT Grant SFRH/BPD/47245/2008. João Leite was partially supported by FCT Project ASPEN PTDC/EIA-CCO/110921/2009. Martin Slota was partially supported by FCT Grant SFRH/BPD/47245/2008.

References

1. Alferes, J.J., Knorr, M., Swift, T.: Queries to hybrid MKNF knowledge bases through oracular tabling. In: The Semantic Web - ISWC 2009. LNCS, vol. 5823, pp. 1–16. Springer (2009)
2. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press (2003)
3. Boella, G., Governatori, G., Rotolo, A., van der Torre, L.: Lex minus dixit quam voluit, lex magis dixit quam voluit: A formal study on legal compliance and interpretation. In: Casanovas, P., Pagallo, U., Sartor, G., Ajani, G. (eds.) AI Approaches to the Complexity of Legal Systems. Complex Systems, the Semantic Web, Ontologies, Argumentation, and Dialogue - International Workshops AICOL-I/IVR-XXIV Beijing, China, September 19, 2009 and AICOL-II/JURIX 2009, Rotterdam,The Netherlands, December 16, 2009 Revised Selected Papers. Lecture Notes in Computer Science, vol. 6237, pp. 162–183. Springer (2009)
4. Boella, G., Governatori, G., Rotolo, A., van der Torre, L.: A logical understanding of legal interpretation. In: Lin, F., Sattler, U., Truszczynski, M. (eds.) KR. AAAI Press (2010)
5. Boella, G., Pigozzi, G., van der Torre, L.: Normative framework for normative system change. In: Sierra, C., Castelfranchi, C., Decker, K.S., Sichman, J.S. (eds.) AAMAS (1). pp. 169–176. IFAAMAS (2009)
6. Boella, G., van der Torre, L., Verhagen, H.: Introduction to normative multiagent systems. Computational and Mathematical Organization Theory 12, 71–79 (2006), http://dx.doi.org/10.1007/s10588-006-9537-7, 10.1007/s10588-006-9537-7
7. Chen, W., Warren, D.S.: Tabled Evaluation with Delaying for General Logic Programs. J. ACM 43(1), 20–74 (January 1996)
8. DeYoung, H., Garg, D., Jia, L., Kaynar, D.K., Datta, A.: Experiences in the logical specification of the HIPAA and GLBA privacy laws. In: Al-Shaer, E., Frikken, K.B. (eds.) WPES. pp. 73–82. ACM (2010)
9. Gelder, A.V., Ross, K.A., Schlipf, J.S.: The well-founded semantics for general logic programs. Journal of the ACM 38(3), 620–650 (1991)
10. Giovanni Sartor, Pompeu Casanovas, M.A.B., Fernandez-Barrera, M. (eds.): Approaches to Legal Ontologies: Theories, Domains, Methodologies. Law, Governance and Technology series, Springer (2011)
11. Gomes, A.S., Alferes, J.J., Swift, T.: Implementing query answering for hybrid mknf knowledge bases. In: PADL. pp. 25–39 (2010)
12. Governatori, G., Rotolo, A.: Bio logical agents: Norms, beliefs, intentions in defeasible logic. Autonomous Agents and Multi-Agent Systems 17(1), 36–69 (2008)

13. Governatori, G., Rotolo, A.: Norm compliance in business process modeling. In: Dean, M., Hall, J., Rotolo, A., Tabet, S. (eds.) Semantic Web Rules - International Symposium, RuleML 2010, Washington, DC, USA, October 21-23, 2010. Proceedings. Lecture Notes in Computer Science, vol. 6403, pp. 194–209. Springer (2010)
14. Knorr, M., Alferes, J.J., Hitzler, P.: A coherent well-founded model for hybrid mknf knowledge bases. In: ECAI. pp. 99–103 (2008)
15. Knorr, M., Alferes, J.J., Hitzler:, P.: Local closed world reasoning with description logics under the well-founded semantics. Artificial Intelligence (2011), accepted for publication
16. Lifschitz, V.: Nonmonotonic databases and epistemic queries. In: Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI'91). pp. 381–386 (1991)
17. Makinson, D., van der Torre, L.W.N.: What is input/output logic? input/output logic, constraints, permissions. In: Boella, G., van der Torre, L.W.N., Verhagen, H. (eds.) Normative Multi-agent Systems. Dagstuhl Seminar Proceedings, vol. 07122. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany (2007)
18. Motik, B., Rosati, R.: Reconciling description logics and rules. Journal of the ACM 57(5) (2010)
19. Motik, B., Sattler, U., Studer, R.: Query answering for owl-dl with rules. J. Web Sem. 3(1), 41–60 (2005)
20. Prakken, H., Sartor, G.: Argument-based extended logic programming with defeasible priorities. Journal of Applied Non-Classical Logics 7(1) (1997)
21. Prakken, H., Sartor, G.: The role of logic in computational models of legal argument: A critical survey. In: Kakas, A.C., Sadri, F. (eds.) Computational Logic: Logic Programming and Beyond. Lecture Notes in Computer Science, vol. 2408, pp. 342–381. Springer (2002)
22. Schmidt-Strauss, M., Smolka, G.: Attributive concept descriptions with complements. Artificial Intelligence 48, 1–26 (1990)
23. Sergot, M.J., Sadri, F., Kowalski, R.A., Kriwaczek, F., Hammond, P., Cory, H.T.: The british nationality act as a logic program. Commun. ACM 29, 370–386 (May 1986), http://doi.acm.org/10.1145/5689.5920
24. Shoham, Y., Tennenholtz, M.: On social laws for artificial agent societies: off-line design. Artif. Intell. 73, 231–252 (February 1995), http://dx.doi.org/10.1016/0004-3702(94)00007-N
25. Slota, M., Leite, J.: Towards closed world reasoning in dynamic open worlds. TPLP 10(4-6), 547–563 (2010)
26. Slota, M., Leite, J., Swift, T.: Splitting and updating hybrid knowledge bases. TPLP (2011), to appear
27. Swift, T., Warren, D.S.: Cold Dead Fish: A System for Managing Ontologies (2003), available via http://xsb.sourceforge.net