Principles for a Judgement Editor Based on Multi-BDDs

Guillaume Aucher[1], Anthony Baire[1], Jean Berbinau, Annie Foret[1], Jean-Baptiste Lenhof[2], Marie-Laure Morin[3], Olivier Ridoux[1], and François Schwarzentruber[2]

[1] University of Rennes 1 guillaume.aucher@irisa.fr
[2] Ecole Normale Supérieure of Rennes jean-baptiste.lenhof@ens-rennes.fr
[3] Cour de cassation mlmorin@orange.fr

Abstract. We describe the theoretical principles that underlie the design of a software tool which could be used by judges for writing judgements and for making decisions about litigations. The tool is based on Binary Decision Diagrams (BDD), which are graphical representations of truth–valued functions associated to propositional formulas. Given a specific litigation, the tool asks questions to the judge; each question is represented by a propositional atom. Their answers, true or false, allow to evaluate the truth value of the formula which encodes the overall recommendation of the software about the litigation. Our approach combines some sort of 'theoretical' or 'legal' reasoning dealing with the core of the litigation itself together with some sort of 'procedural' reasoning dealing with the protocol that has to be followed by the judge during the trial: some questions or group of questions must necessarily be examined and sometimes in a specific order. That is why we consider extensions of BDDs called Multi-BDDs. They are BDDs with multiple entries corresponding to the different specific issues that must necessarily be addressed by the judge during the trial. We illustrate our ideas on a case study dealing with French union trade elections, an example that has been used throughout a project with the French Cour de cassation. We end the article by sketching the architecture of a prototype software that has been developed during this project.

1 Introduction

Law has so much impact on our daily lives that any mistake related to its application must be avoided. Law must be transparent, accountable and understandable by anybody it can affect. Clearly, computer science can be instrumental to reach these goals, in particular because French law (stemming from Roman law) is organized in a systematic manner and lends itself easily to a formalization under the form of rules and principles which are very common in the models used in computer science. Logic, sometimes viewed as the "calculus of computer science" [15], is a natural theoretical background to address these issues. As it turns out, there is already an important body of work in the field of logic and law (see [7,21,14] for details and pointers).

1.1 A Status Quo in Logic and Law

Originally, logic was intended to be used for the representation of law in a clear and unambiguous manner. On top of this representation, some kind of reasoning could then take place to infer some information. Sergot and Kowalski were pioneers in the use of logic programming in that field, which they applied to the formalization of the British Nationality Act [22]. However, they encountered difficulties with the Prolog treatment of negation as failure. In Prolog, something is said to be false if it is not known or cannot be inferred to be true. Hence, if a program cannot show that an infant was born in the UK, it assumes that it was not. But more generally, researchers realized that several aspects of the law which can not be dealt with standard (Fregean) logic had to be taken into account, such as the need to handle exceptions, conflicting rules, vagueness, open texture (i.e. the failure of natural languages to determine future usage of terms), counterfactual conditionals and the possibility of rational disagreement. This led to the development of a broader conception of logic, which is in fact a special case of a more general and relatively recent development of logic where the central focus is the study of rational agency and intelligent interaction [18].

All this said, a striking particularity of most of the works which have been pursued at the interface of logic and law in the last decades is that they were mostly driven by theoretical considerations and without much interaction with jurists and lawyers. Arguably, this work did not really catch the attention of the lawyers and jurists. In particular, they did not change the way they work or their actual practice of the law, except maybe for the adoption of large and online databases such as LexisNexis or Legifrance[4] (based on standards for legal documents such as LegalRuleXML) and the use of knowledge management systems [8]. This theoretical work did not seem for jurists to answer an actual need and it was somehow remote from their daily preoccupations, although the researchers could sense the potential and the important applicability of their work in the practice of the law.

1.2 Current Problems in the Application of Law in France

The work that we are going to describe in this article stems from actual problems expressed to us by jurists of the Cour de cassation, the highest legal institution in France.[5] These problems are in fact not specific to France. The application of law is plagued with a series of problems which are difficult to overcome with the standard and present methods employed by jurists. First, the increasing diversity and complexity of legal texts and jurisprudence makes the work of jurists (and lawyers) very difficult to pursue nowadays. This complexity appears not only

[4] See `www.lexisnexis.com` and `https://www.legifrance.gouv.fr/`.

[5] The main role of the "Cour de cassation" is to check that the law is applied correctly and uniformly in France, mainly from a formal point of view. The 'Cour de cassation' is also the last legal institution to which French citizens can resort to in case they want to 'break' ('casser' in French) a legal decision that concerns them. In that case, the decision of the Cour de cassation cannot be revoked.

at the local or national level but is sometimes worsened by its interaction with the european level, and sometimes even the international level. Second, legal texts and jurisprudence are changing at a high pace in some areas and it is difficult for jurists (and lawyers) to cope and keep up-to-date with the current legislation and regulations. Third, there is a lack of uniformity in the application of law, depending on the geographical part in which trials take place, on the local customs and sometimes the personnality of the judges, and more generally on the specific political or social context in which a legal decision is taken. Altogether, these three problems call for a new kind of solution.

1.3 A New Kind of Solution: a Software Assistant

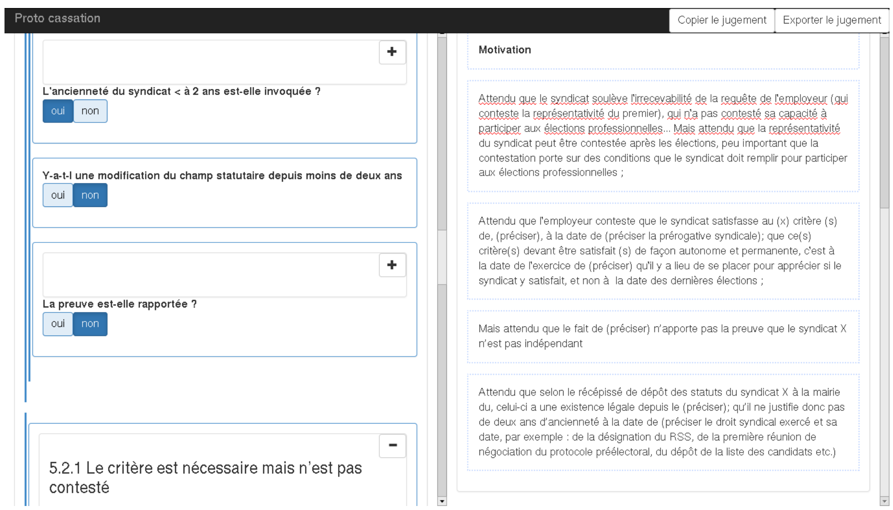


Fig. 1. Screenshot of the graphical user interface

The solution propounded by the Cour de cassation is to make use of a software, whose ultimate role is to help judges write a judgement and take better and well-informed decisions thanks to a series of questions to which she/he has to answer. These questions would be backed up by the corresponding legal texts and jurisprudence.

This software assistant would indeed be a solution to the problems mentionned above. First, it would unify and uniformize the application of law in France: the kind of reasoning proposed by the software to sort out a given (type of) litigation could be controlled by the Cour de cassation and it could also be the same in every jurisdiction of France. Second, like any software, it could take into account the evolution of legal texts and jurisprudence to update the different kinds of reasoning and therefore cope with the increasing complexity of law.

Third, its easy access to a large and up-to-date knowledge base comprising the current legal texts and jurisprudence would increase the chance for the judge to make well–informed decisions.

The graphical user interface (GUI) of this software is depicted in Figure 1. On the left hand side of the GUI, a guide for reasoning consisting of a series of questions is displayed. These questions and their underlying reasoning are backed up by legal texts and jursiprudence to which the user can have access whenever she/he wants. On the right hand side of the GUI, a judgement is produced automatically as the user answers the questions. The user can modify at any time the judgement produced and can also have an alternative graphical representation of the web of questions to which she/he has to answer on the left hand side.

1.4 Structure of the Article

The article is organized as follows. In Section 2, we recall the basics of propositional logic and BDDs. In Section 3, we extend propositional logic with the examination operator $!\varphi$ and we provide a semantics to this operator by means of Multi-BDDs. In Section 4, we consider as case study the problems of determining whether an association of employees in a firm can indeed be considered ('qualified') as a trade union. In Section 5, we show how the various algorithms that have been designed for BDDs can be used to solve and address specific kinds of legal issues that arise in the practice of the law. Finally, we conclude in Section 6.

2 Propositional logic and BDD

In this section, we recall the basics of propositional logic and Binary Decision Diagrams (BDD for short, [9]) and we show how they are related to each other. BDD can be viewed as an operational semantics of propositional logic and it is this *operational* feature that will play a role in the legal context. Indeed, it will allow us to represent the *procedural* aspect of the practice of law (during a trial especially).

2.1 Propositional Logic

In the sequel, $\mathbb{P}$ is a set of *atoms* (propositional letters) denoted $p, q, r, \ldots$ and T and F are two symbols called *truth values* standing for True and False.

Definition 1 (Propositional language $\mathcal{L}$). *The language $\mathcal{L}$ is the set that contains $\mathbb{P} \cup \{\top, \bot\}$ and such that*

- *if $\varphi, \psi \in \mathcal{L}$, then $\neg\varphi, (\varphi \wedge \psi), (\varphi \vee \psi), (\varphi \to \psi) \in \mathcal{L}$;*
- *$\mathcal{L}$ contains no more formulas.*

We introduce the following abbreviation: $\varphi \leftrightarrow \psi \triangleq (\varphi \to \psi) \wedge (\psi \to \varphi)$*. The formula* $\varphi[p\backslash\psi]$ *denotes the formula* φ *where the atom* p *is uniformly substituted with* ψ*.*

The intuitive reading of the formulas is as follows: $\neg\varphi$: "φ does not hold"; $\varphi \wedge \psi$: "φ holds and ψ holds"; $\varphi \vee \psi$: "φ holds or ψ holds"; $\varphi \to \psi$: "If φ holds then ψ holds".

Definition 2 (Interpretation). *A* total (partial) interpretation *is a total (resp. partial) function* $\mathcal{I} : \mathbb{P} \mapsto \{T, F\}$ *that assigns one of the* truth values T *or* F *to* every *(resp.* some *of the) atoms in* $\mathbb{P}$*. The set of total interpretations is denoted* $\mathcal{C}$ *and the set of partial interpretations is denoted* $\mathcal{C}^p$*. Note that* $\mathcal{C} \subseteq \mathcal{C}^p$*. If* $\mathcal{I} \in \mathcal{C}^p$*, then* $Ext(\mathcal{I})$ *is the set of* total *interpretations extending the interpretation* $\mathcal{I}$*, that is, for all* $\mathcal{I}' \in Ext(\mathcal{I})$*, for all* $p \in \mathbb{P}$ *such that* $\mathcal{I}(p)$ *is defined, we have that* $\mathcal{I}(p) = \mathcal{I}'(p)$*.*

We can extend the domain of an interpretation function from the set of atoms to the set of all formulas of $\mathcal{L}$*. This extension is inductively defined by the truth table given in Figure 2. If* E *is a set of interpretations, we say that a formula* φ *of* $\mathcal{L}$ *is* valid on E *when for all* $\mathcal{I} \in E$*, we have that* $\mathcal{I}(\varphi) = T$*. When* $E = \mathcal{C}$*, we simply say that* φ *is* valid*.*

$\mathcal{I}(\varphi)$	$\mathcal{I}(\psi)$	$\mathcal{I}(\neg\varphi)$	$\mathcal{I}(\varphi \wedge \psi)$	$\mathcal{I}(\varphi \to \psi)$	$\mathcal{I}(\varphi \vee \psi)$
T	T	F	T	T	T
T	F	F	F	F	T
F	T	T	F	T	T
F	F	T	F	T	F

Fig. 2. Semantics of Propositional Connectives

2.2 Ordered Binary Decision Diagrams (OBDD)

Definition 3 (Binary Decision Diagram, BDD, [6]). *A* binary decision diagram (BDD) *is a directed acyclic graph with a unique root, which is also called the* entry point*. Each leaf is labeled with a constant* T *or* F*. Each interior node is labeled with an atom and has two outgoing edges: one, the* false edge*, is denoted by a dotted line, while the other, the* true edge*, is denoted by a solid line. No atom appears more than once in a branch from the root to an edge.*

Example 1. Figure 3 shows a binary decision diagram.

During a trial, some questions have to be examined in a certain temporal order. This temporal order does not play a role from a logical point of view, in the sense that the truth value of a given statement will not depend on the order

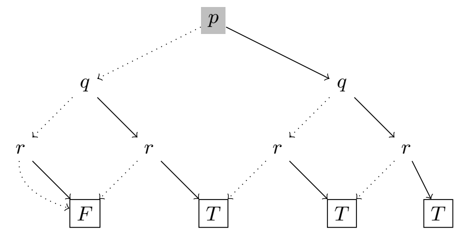


Fig. 3. Example of a binary decision diagram

in which its arguments are examined. However, this temporal order plays a role from a procedural point of view when the judge constructs its judgment while answering the different questions.

This ordering is made explicit in *ordered* binary decision diagrams (OBDD): we can canonically associate to each OBDD an ordering corresponding to the order in which the different questions should be examined by the judge.

Definition 4 (Compatible set of orderings). *Let $\mathcal{P}$ be a finite set of atoms or propositional constants. Let $\mathcal{O} = \{(\mathcal{O}^1, <^1), \ldots, (\mathcal{O}^n, <^n)\}$, where for each i, $\mathcal{O}^i$ is a set of elements of $\mathcal{P}$ ordered by a total relation $<^i$. $\mathcal{O}$ is a* compatible set of orderings for $\mathcal{P}$ *iff for all $i \neq j$, there are no atoms $p, p' \in \mathcal{O}^i \cap \mathcal{O}^j$ such that $p <^i p'$ while $p' <^j p$.*

Definition 5 (Ordered Binary Decision Diagrams, OBDD).

- *Let bdd be a BDD. The* set of orderings associated to *bdd is the set $\{(\mathcal{O}^1, <^1), \ldots, (\mathcal{O}^n, <^n)\}$ where each set corresponds to the atoms appearing in the i^{th} branch of the bdd and the ordering $<^i$ is defined by the order of appearance of each atom on the i^{th} branch (starting from the root of the branch).*
- *An* ordered binary decision diagram (OBDD) *is a BDD such that the set of orderings associated to this BDD is compatible.*

Note that the set of orderings associated to an OBDD induces an ordering on *all* the atoms appearing in the OBDD.

Definition 6 (Operations on OBDDs, [6]). *The operations `Apply`, `Restrict` and `Reduce` are defined in Figure 4. Two BDDs bdd and bdd′ are said to be* equivalent, *written $bdd \equiv bdd'$, when `Reduce`(bdd)=`Reduce`(bdd').*

Theorem 1 ([9]). *For all BDDs bdd, `Reduce`$(bdd) \equiv bdd$.*

Instead of the set-theoretical semantics of propositional logic based on the notion of interpretation, we can provide a semantics to propositional logic in terms of OBDDs. The meaning of a propositional formula is completely determined by the OBDD associated to that formula, which is itself built inductively from the `Apply` Algorithm. The soundness of `Apply` is ensured by Shannon's expansion Theorem:

1. Algorithm `Restrict`:

   Input: An OBDD *bdd* for a formula φ; an interpretation $\mathcal{I} \in \mathcal{C}^p$ (partial or total).

   Output: An OBDD $\texttt{Restrict}(\mathcal{I}, bdd)$ for a formula ψ such that $\varphi \leftrightarrow \psi$ is valid on the set of interpretations $Ext(\mathcal{I})$.

   Perform a recursive traversal of the OBDD:

   – If the root of *bdd* is a leaf, return the leaf.
   – If the root of *bdd* is labeled p and $\mathcal{I}(p)$ is defined, return the sub-BDD reached by its true edge if $\mathcal{I}(p) = T$ and the sub-BDD reached by its false edge if $\mathcal{I}(p) = F$.
   – Otherwise (the root of *bdd* is labeled p and $\mathcal{I}(p)$ is *not* defined), apply the algorithm to the left and right sub-BDDs, and return the BDD whose root is p and whose left and right sub-BDDs are those returned by the recursive calls.

2. Algorithm `Apply`:

   Input: OBDDs bdd_φ for formula φ and bdd_ψ for formula ψ such that the set of orderings $\{(\mathcal{O}_\varphi, <_\varphi), (\mathcal{O}_\psi, <_\psi)\}$ is compatible (where $(\mathcal{O}_\varphi, <_\varphi)$ and $(\mathcal{O}_\psi, <_\psi)$ are the orderings associated to bdd_φ and bdd_ψ respectively). A connective $\star \in \{\neg, \wedge, \vee, \rightarrow\}$.

   Output: An OBDD for the formula $\varphi \star \psi$, denoted $bdd_\varphi \star bdd_\psi$, if $\star \in \{\wedge, \vee, \rightarrow\}$, or for the formula $\neg\varphi$, denoted $bdd_{\neg\varphi}$.

   Perform a recursive traversal of both OBDDs as follows.

   – If $\star = \neg$, then return bdd_φ where the leafs labeled T are replaced by leafs labeled F, and vice versa.
   – Otherwise ($\star$ is then a binary connective)
     • If the root of bdd_φ or bdd_ψ is a leaf, then return either T, F, the other BDD, or `Apply`(the other BDD, $\neg$), according to the truth table of $\star$.
     • Otherwise select the bdd_φ or bdd_ψ that have the root p of lowest rank in the atom ordering, apply recursively the algorithm to its left and right sub-BDDs, with the other BDD as other parameter, and return the BDD whose root is p and whose left and right sub-BDDs are those returned by the recursive calls.

3. Algorithm `Reduce`:

   Input: An OBDD *bdd*.

   Output: A reduced OBDD bdd'.

   Perform a recursive traversal of the OBDD:

   – If *bdd* has more than two distinct leaves (one labeled with T and one labeled with F), remove duplicate leaves. Direct all edges that pointed to leaves to the remaining two leaves.
   – Perform the following steps as long as possible:
     (a) If all outgoing edges of a node labeled p_i point at the same node labeled p_j, delete this node for p_i and direct p_i's incoming edges to p_j.
     (b) If two nodes labeled p_i are the roots of identical sub-BDDs, delete one sub-BDD and direct its incoming edges to the other node.

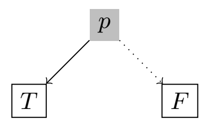
Fig. 4. Schematic algorithms `Restrict`, `Apply` and `Reduce`.

Theorem 2 (Shannon expansion, [6]). *For all formulas $\varphi, \psi \in \mathcal{L}_M$, for all $\star \in \{\wedge, \vee, \rightarrow\}$, the following formula is valid:*

$$\varphi \star \psi \leftrightarrow (p \wedge (\varphi[p\backslash\top] \star \psi[p\backslash\top])) \vee (\neg p \wedge (\varphi[p\backslash\bot] \star \psi[p\backslash\bot]))$$

Definition 7 (OBDD associated to a formula). *Let $\varphi \in \mathcal{L}$. The OBDD associated to φ, written bdd_φ, is defined inductively on φ as follows:*

- *$bdd_\top$ is the OBDD consisting of a single node labeled T;*
- *$bdd_\bot$ is the OBDD consisting of a single node labeled F;*
- *bdd_p is the following OBDD:*



- *$bdd_{\varphi\star\psi}$ is the OBDD $bdd_\varphi \star bdd_\psi$ obtained from Algorithm `Apply` of Figure 4, for all $\star \in \{\wedge, \vee, \rightarrow\}$.*

3 The Examination Operator

Sometimes during a trial, the judge must necessarily examine or raise a specific issue. For example, the complainant can attack the legitimity of a syndicate that presents a candidate to the professionnal elections of a firm. The complainant could argue that this association of employees cannot really be qualified as a syndicate because it is not senior enough. In that case, even if the syndicate turns out to be senior enough (older than 2 years of existence), the judge *must* nevertheless examine *all* the other criteria (different from seniority) that make the association of employees qualify as a syndicate (even if she/he does not decide to raise the issue of the other criteria during the trial).

Definition 8 (Language $\mathcal{L}_M$). *The language $\mathcal{L}_M$ is the set that contains $\mathbb{P} \cup \{\top, \bot\}$ and such that*

- *if $\varphi, \psi \in \mathcal{L}_M$, then $!\varphi, \neg\varphi, (\varphi \wedge \psi), (\varphi \vee \psi), (\varphi \rightarrow \psi) \in \mathcal{L}_M$;*
- *$\mathcal{L}_M$ contains no more formulas.*

We use the same abbreviation and notation as in Definition 1.

The intuitive reading of the formula $!\varphi$ is as follows: "φ is examined and it holds". For example, the formula $q \wedge !p$ holds if, and only if, p is examined and p and q both hold.

We must provide a semantics to our extended language $\mathcal{L}_M$, in particular to the examination operator $!\varphi$. The Algorithm `Apply` provides already a semantics to every formula of the propositional language in terms of OBDD. Indeed, it suffices to apply it inductively to every subformula of a given formula φ (from the atoms up to the formula φ) to obtain an OBDD that captures the meaning of φ. Thus, we extend this algorithm to cover the examination operator. This leads us to introduce an extended form of BDDs with several entry points. This extension is called a *Multi-BDD* and, contrary to BDDs, it can also have several roots.

Definition 9 (Multi-BDD). *A* Multi-BDD *(MBDD for short) is a directed acyclic graph with the same property as a BDD except that it can have multiple roots. The set of* entry points *of a MBDD is a set of nodes comprising the roots of the MBDD and possibly others.*

Example 2. Figure 7 shows a MBDD. The construction of this MBDD is detailed in the next section.

Definition 10 (MBDD associated to a formula). *Let $\varphi \in \mathcal{L}_M$. The MBDD associated to φ, written $mbdd_\varphi$, is defined inductively on φ as follows. Let $\star \in \{\wedge, \vee, \rightarrow\}$.*

- *$mbdd_\top$, $mbdd_\bot$, $mbdd_p$ are respectively $bdd_\top$, $bdd_\bot$, bdd_p of Definition 7.*
- *if $\varphi \neq !\varphi'$ and $\psi \neq !\psi'$ (for any $\varphi', \psi' \in \mathcal{L}_M$), then $mbdd_{\varphi\star\psi}$, is the MBDD $mbdd_\varphi \star mbdd_\psi$ obtained from Algorithm* `Apply` *of Figure 4;*
- *if $\varphi = !\varphi'$ and $\psi \neq !\psi'$ (for some $\varphi' \in \mathcal{L}_M$ and any $\psi' \in \mathcal{L}_M$) then $mbdd_{\varphi\star\psi}$ is the disjoint union of the MBDDs $mbdd_\varphi$ and $mbdd_{\varphi\star\psi}$. Hence, the set of entry points of $mbdd_{!\varphi\star\psi}$ is the union of the entry points of $mbdd_\varphi$ and $mbdd_{\varphi\star\psi}$;*
- *if $\varphi = !\varphi'$ and $\psi = !\psi'$ (for some $\varphi' \in \mathcal{L}_M$ and some $\psi' \in \mathcal{L}_M$) then $mbdd_{\varphi\star\psi}$ is the disjoint union of the MBDDs $mbdd_\varphi$, $mbdd_\psi$ and $mbdd_{\varphi\star\psi}$. Hence, the set of entry points of $mbdd_{!\varphi\star\psi}$ is the union of the entry points of $mbdd_\varphi$, $mbdd_\psi$ and $mbdd_{\varphi\star\psi}$.*

In the same spirit, we could also define the notion of *Ordered* Multi-BDD, which might be even more suited to deal with the procedural reasoning of legal practice. It is important to notice that in order to reduce a MBDD to T or F, the user/judge must evaluate *every* formula that corresponds to the sub-BBD generated by an entry point of the MBDD. The theorem below shows that our notion of MBDD does capture that: if $\texttt{Restrict}(\mathcal{I}, mbdd_\varphi) \in \{T, F\}$, then every occurrence of a subformula $!\psi$ in φ is such that $\texttt{Restrict}(\mathcal{I}, mbdd_\psi) \in \{T, F\}$.

Theorem 3. *Let φ be a formula of $\mathcal{L}_M$ containing a subformula $!\psi$ and let $\mathcal{I} \in \mathcal{C}^p$ be a partial interpretation. If $\mathcal{I}(\psi) \notin \{T, F\}$, then $\texttt{Restrict}(\mathcal{I}, mbdd_\varphi) \notin \{T, F\}$.*

Example 3. In Figure 5, we represent the MBDD for the formula $q \wedge !p$ (third graph). This formula is true if, and only if, p and q are both true and p is examined. Hence, if q is given the value F (by the judge), then the truth value of the formula $q \wedge !p$ is still not determined. Indeed, the MBDD is still not completely reduced and we must *examine* the 'question' p and give a truth value to p.

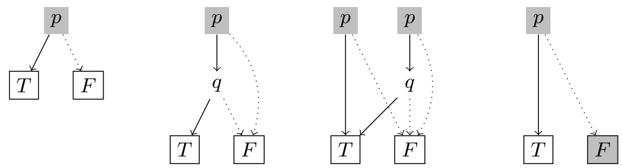


Fig. 5. bdd_p (*first*), $bdd_{q \wedge p}$ (*second*), $mbdd_{q \wedge !p}$ reduced (*third*), $\texttt{Restrict}(\mathcal{I}, mbdd_{q \wedge !p})$ reduced (*fourth*) where $\mathcal{I}(q) = F$ and $\mathcal{I}(p)$ is undefined.

4 Case Study: French Union Trades

Our case study deals with French professionnal election in a firm [23]. The problem is to determine whether an association of employees is really qualified as a union so that this association can indeed propose candidates to professionnal elections in the firm. Law introduces four criteria that have to be fulfilled so that an association can indeed be qualified as a union. These four criteria have to hold altogether, and during a trial, the judge must necessarily examine all of them. They are the following:

1. the association of employees should respect the 'Republican values';
2. the association of employees should be 'Independent' (from the directorate of the firm for example);
3. the association of employees should be 'Senior' enough (minimum 2 years of existence);
4. the association of employees should be within the appropriate 'Geographical and professionnal' range.

Hence, we introduce the formula R (resp. I, S and G) which stands respectively for "the criteria of the *Republican values* (resp. *Independence*, *Seniority*, *Geographical and professionnal range*) is fulfilled". The four criteria must hold for an association of employees to be legally qualified as union in a firm, and they must all be examined by the judge, even if they are not contested. Therefore, the following formula must be true: $!R \wedge !I \wedge !G \wedge !S$.

4.1 The Criterias of 'Republican values' and 'Independence'

To determine whether the criteria of 'Republican values' holds, the judge has to answer a number of questions. To formulate these questions, we introduce the following set of atoms:

$$\mathbb{P}_R \triangleq \{Lit_R, OldJug_R, NewElt_R, Proof_{\neg R}\}$$

These atoms stand for the following propositions:

- Lit_R: "The criteria of republican values is contested";

- $OldJug_R$: "An old judgement dealing with 'Republican values' has already established that the association of employees fulfills the criteria of Republican Values";
- $NewElt_R$: "New elements have been brought to the fore that oblige to reconsider the old judgement";
- $Proof_{\neg R}$: "The complainant presents the proof that the criteria of 'Republican values' is not fulfilled".

Then, we can give the formula in the language $\mathcal{L}_M$ that determine in which case the criteria of 'Republican values' holds. It is the following:

$$R \triangleq Lit_R \rightarrow ((OldJug_R \wedge \neg NewElt_R) \vee (\neg OldJug_R \wedge \neg Proof_{\neg R}))$$

The above formula reads as follows: "if the plaintive contests that the criteria of republican value is satisfied, then either there is an old judgement that already established that this criteria was fulfilled and no new element has been brought to the fore that oblige to reconsider this old judgement, or there is no old judgement and the complainant has not been able to prove that the criteria is not fulfilled".

Dealing with the criteria of 'Independence' is completely similar. Hence, we introduce the following set of atoms:

$$\mathbb{P}_I \triangleq \{Lit_I, OldJug_I, NewElt_I, Proof_{\neg I}\}$$

Their interpretation is the same as for the criteria of Republican values, except that the term "Republican values" has to be replaced by "Independence" everywhere. So, the formula in the language $\mathcal{L}_M$ that determines in which case I holds is the following:

$$I \triangleq Lit_I \rightarrow ((OldJug_I \wedge \neg NewElt_I) \vee (\neg OldJug_I \wedge \neg Proof_{\neg I}))$$

Its intuitive interpretation is the same as for the previous criteria.

Three equivalent representations. In Figure 6, we give two equivalent and alternative representations of the semantics of the formula R (in propositional logic): the truth table of R and the MBDD associated to R.

4.2 The Criteria of 'Geographical and Professional Range'

To determine whether the criteria of "Geographical and professional range" holds, the judge has to answer a number of questions. To formulate these questions, we introduce the following set of atoms:

$$\mathbb{P}_G \triangleq \{Lit_G, Decide_G, Proof_{\neg G}\}$$

These atoms stand for the following propositions:

- Lit_G: "The criteria of Geographical and professionnal range is contested";
- $Decide_G$: "The judge decides to examine the criteria of Geographical and professionnal range";

– $Proof_{\neg G}$: "The complainant presents the proof that the criteria of 'Geographical and professional range' is not fulfilled".

Then, we can give the formula in the language $\mathcal{L}_M$ that determine in which case the criteria of 'Geographical and professional range' holds. It is the following:

$$G \triangleq (Lit_G \vee Decide_G) \rightarrow Proof_{\neg G}$$

The above formula reads as follows: "if the complainant contests that the criteria of 'Geographical and professional range' is fulfilled or if the judge decides to consider this criteria, then the complainant presents the proof that the criteria is not fulfilled". The MBDD associated to the formula G is depicted in Figure 7 (*left*), it is the third OBDD from the left.

4.3 A MBDD for Qualifying as a Union Trade

We have not considered the criteria of 'Seniority' so far and we will not deal with it in order to ease the presentation and because it is more complex to represent than the other criteria.

The MBDD for $!R \wedge !I \wedge !G$ is given in Figure 7. It is simply the disjoint union of the three OBDDs R, I, G and $R \wedge I \wedge G$. Then, it can be reduced equivalently thanks to the Algorithm `Reduce` of Figure 4. This yields the MBDD on the left of Figure 7. Note that in this MBDD, all the leaf nodes have been merged into two nodes ($bdd_\top$ and $bdd_\bot$) and the OBDD for the formula G has been merged with the OBDD for the formula $R \wedge I \wedge G$. That is why we have an entry node Lit_G which is not a root of the MBDD.

5 Applications of BDD Algorithms to Legal Reasoning

Because our solution is based on BDDs, we inherit from the vast amount of work for BDDs a number of algorithms and software applications that can play an important role in legal reasoning. Even if they were not initially intended to be used in the legal domain, these algorithms and software applications turn out to be really relevant for solving specific problems or answer specific queries of the judge. We list below some of these algorithms (some of them have already been considered above) and show how they can be used by a judge during, before or after a trial. We start with the algorithms of this article:

– Algorithm `Restrict`. This algorithm is used when the judge answer questions: each question answered corresponds in fact to an application of the algorithm `Restrict`. After each answer, the MBDD is instanciated and the node corresponding to the question disappears.
– Algorithm `Reduce`. This algorithm can be used to determine whether two kinds of legal reasoning represented by two different MBDDs are in fact equivalent: in case the MBDD returned by this algorithm is the same in both cases, then they are indeed equivalent (see Theorem 1).

$\mathcal{I}(Lit_R)$	$\mathcal{I}(OldJug_R)$	$\mathcal{I}(NewElt_R)$	$\mathcal{I}(Proof_{\neg R})$	$\mathcal{I}(R)$
T	T	T	T	F
T	T	T	F	T
T	T	F	T	T
T	T	F	F	T
T	F	T	T	F
T	F	T	F	T
T	F	F	T	F
T	F	F	F	T
F	T	T	T	T
F	T	T	F	T
F	T	F	T	T
F	T	F	F	T
F	F	T	T	T
F	F	T	F	T
F	F	F	T	T
F	F	F	F	T

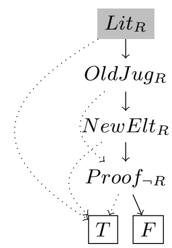


Fig. 6. Two equivalent and alternative semantics for the formula $Lit_R \rightarrow ((OldJug_R \wedge \neg NewElt_R) \vee (\neg OldJug_R \wedge \neg Proof_{\neg R}))$.

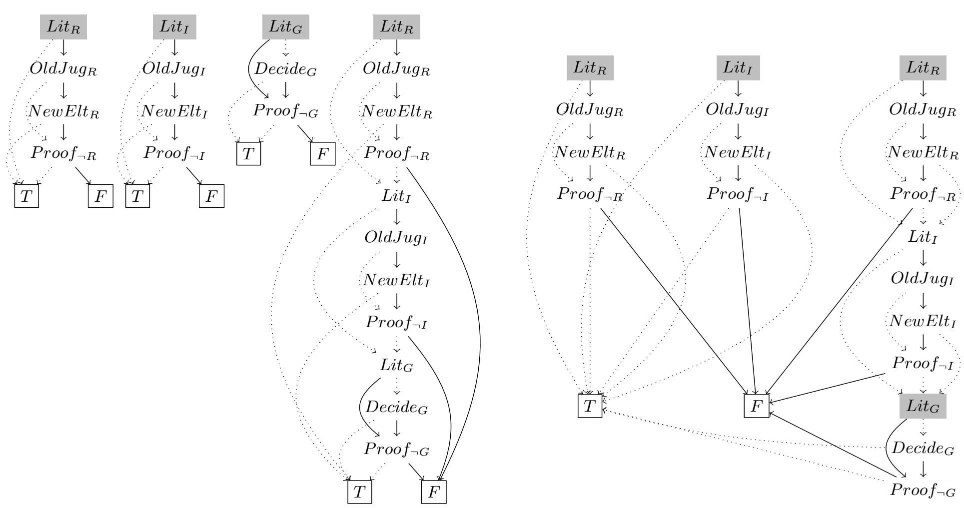


Fig. 7. MBDD for $!R \wedge !I \wedge !G$ (*left*) and MBDD for $!R \wedge !I \wedge !G$ reduced (*right*). It has four entry points (in gray) corresponding to R, I, G and $R \wedge I \wedge G$.

– Algorithm `Apply`. This algorithm can be used to construct a MBDD corresponding to a formula expressed in our language $\mathcal{L}_M$. It can also be used when we want to combine two kinds of legal reasoning that deal with different but complementary issues that have already been represented by two MBDDs.

Other algorithms based on BDDs dealing with quantification over propositional atoms can be used to solve the following tasks:

– Determine whether the answer to a specific question will allow the judge to conclude about a litigation or a specific subproblem without having to examine all the other questions exhaustively.
– Determine whether a question is redondant and can thus be removed from the MBDD.

These algorithms are only a few among the large amount of algorithms for BDDs that are available. Many other algorithms can be used and even designed to solve specific legal reasoning tasks.

6 Conclusion

The ideas and principles presented in this article have been implemented in a prototype in the course of our project with the French Cour de cassation. This prototype was tried out by three judges of 'cour d'instances' in France. They were all impressed and enthusiastic about the prototype tool that they tested on our case study. Moreover, we also made an experiment with law students of the Ecole Normale Supérieure of Rennes to determine whether they prefer to write the kind of meta-regulations that we use in the software tool with formulas of (propositional) logic or with a graph-based representation like BDDs. We designed a kind of experimental protocol to figure this out. Even if the results were hard to interpret because they had no prior teaching in logic or graph theory, it turns out that they had somehow more facility to use the graph-based representation than the formula-based representation.

We believe that our solution is the most promising and realistic approach to answer the needs expressed to us by the jurists of the Cour de cassation. First, we believe that it is the most rigorous, accurate and solid approach: it is based on methods and techniques of logic that are very well understood, worked out and applied. Our approach allows an important control over our representation of the legal reasoning and over the changes that we may want to make to this reasoning. Second, BDDs are very well–studied and their associated algorithms are able to scale–up to a large number of nodes. Thus, using BDDs is clearly a realistic solution to deal with the complexity of the law and the large amount of texts and jurisprudences. Third, our approach is very flexible and it allows to take into account the evolubility and dynamicity of the law. Indeed, because it is based on propositional logic, the various changes in the law, such as abrogation and annulment, can be modeled naturally as update operations in propositional logic (Governatori & Al. [13,12] attempt to model abrogation and annulment in a more realistic fashion). As it turns out, dealing with dynamism and change has been at the core of most of the recent development in logic and artificial intelligence in the last decades (see for instance [10,24,25]) and many extensions of propositional logic with dynamic operators have been introduced. Hence, the dynamic character of the law could be dealt with in the software by importing, adapting and implementing the various methods and techniques that have been developed in logic for dealing with dynamism and change. This is possible because our approach is based on propositional logic, the most elementary logic.

The tool proposed is only the first part of the whole project since this software would only *use* the MBDDs representing the legal reasoning underlying specific litigations. But these MBDDs would first need to be defined and created by a legal expert or a legislator. The second tool that complements the software described in this article still has to be specified precisely. It should be able to create and edit these MBDDs that represent different forms of legal reasonings. Thus, it should provide mechanisms for modifying the graphs that underly the BDDs. As it turns out, graph modifications and change is also an area of research that is currently very active in logic [3,11,2,5]. These works can be intrumental in checking and verifying that a particular change or modification of the MBDDs has indeed been made and that these changes do correspond to the idea and the intention of the user/legislator who triggered them.

Finally, if such kinds of softwares would ever be developed and used by judges, this would entail as a prerequisite that the jurists be trained and taught some rudiments of logic, especially the law experts who would have to create and edit the MBDDs representing different kinds of legal reasoning. This would also call for the enactement of regulations to determine in which kind of legal context and litigations these softwares can and should be used. Indeed, the novelty of such softwares and their impact on society would raise a number of ethical issues that would need to be harnessed by the law.

References

1. C.E. Alchourrón and E. Bulygin. *Normative systems.* Springer-Verlag WienNew York, 1971.
2. Carlos Areces, Raul Fervari, and Guillaume Hoffmann. Relation-changing modal operators. *Logic Journal of the IGPL*, 23(4):601–627, 2015.
3. Guillaume Aucher, Philippe Balbiani, Luis Fariñas Del Cerro, and Andreas Herzig. Global and local graph modifiers. In *Methods for Modalities 5 (M4M-5)*, Cachan, France, 2007. ENTCS, Elsevier.
4. C. Baier and J.P. Katoen. *Principles of model checking.* MIT press, 2008.
5. Philippe Balbiani, Rachid Echahed, and Andreas Herzig. A dynamic logic for termgraph rewriting. In *ICGT*, pages 59–74, 2010.
6. Mordechai Ben-Ari. *Mathematical logic for computer science.* Springer Science & Business Media, 2012.
7. T. Bench-Capon and H. Prakken. Introducing the logic and law corner. *Journal of logic and computation*, 18(1):1–12, 2008.

8. Guido Boella, Llio Humphreys, Marco Martin, Piercarlo Rossi, and Leendert van der Torre. Eunomos, a legal document and knowledge management system to build legal services. In *International Workshop on AI Approaches to the Complexity of Legal Systems*, pages 131–146. Springer, 2011.
9. Randal E Bryant. Graph-based algorithms for boolean function manipulation. *Computers, IEEE Transactions on*, 100(8):677–691, 1986.
10. Peter Gärdenfors. *Knowledge in Flux (Modeling the Dynamics of Epistemic States)*. Bradford/MIT Press, Cambridge, Massachusetts, 1988.
11. Patrick Girard, Jeremy Seligman, and Fenrong Liu. General dynamic dynamic logic. In *Advances in Modal Logic*, pages 239–260, 2012.
12. Guido Governatori, Vineet Padmanabhan, Antonino Rotolo, and Abdul Sattar. A defeasible logic for modelling policy-based intentions and motivational attitudes. *Logic Journal of the IGPL*, 17(3):227–265, 2009.
13. Guido Governatori and Antonino Rotolo. Changing legal systems: legal abrogations and annulments in defeasible logic. *Logic Journal of the IGPL*, 18(1):157–194, 2010.
14. D. Grossi and A. Rotolo. *A New Survey of Active Directions in Modern Logic*, volume 30 of *Studies in Logic*, chapter Logic in the Law: A Concise Overview, pages 251–274. College Publications London, 2011.
15. J.Y. Halpern, R. Harper, N. Immerman, P.G. Kolaitis, M.Y. Vardi, and V. Vianu. On the unusual effectiveness of logic in computer science. *The Bulletin of Symbolic Logic*, 7(2):213–236, 2001.
16. A.J.I. Jones and M. Sergot. Deontic logic in the representation of law: Towards a methodology. *Artificial Intelligence and Law*, 1(1):45–64, 1992.
17. L.T. McCarty. A language for legal discourse i. basic features. In *Proceedings of ICAIL*, pages 180–189. ACM, 1989.
18. Lawrence S Moss. Applied logic: A manifesto. In *Mathematical problems from applied logic I*, pages 317–343. Springer, 2006.
19. H. Prakken. Formal systems for persuasion dialogue. *The Knowledge Engineering Review*, 21(2):163–188, 2006.
20. H. Prakken and G. Sartor. The role of logic in computational models of legal argument: a critical survey. *Computational Logic: Logic Programming and Beyond*, pages 175–188, 2002.
21. Edwina L. Rissland. *A companion to cognitive science*, chapter Legal Reasoning, pages 722–733. Wiley-Blackwell, 1999.
22. M.J. Sergot, F. Sadri, R.A. Kowalski, F. Kriwaczek, P. Hammond, and H.T. Cory. The british nationality act as a logic program. *Communications of the ACM*, 29(5):370–386, 1986.
23. Yves Struillou, Marie-Laure Morin, and Laurence Pécaut-Rivolier. *Le guide des élections professionnelles 2016-2017 et des désignations de représentants syndicaux dans l'entreprise*. Number 3. Dalloz / Guides Dalloz, 11 2015.
24. Johan van Benthem. *Exploring logical dynamics*. CSLI publications Stanford, 1996.
25. Johan van Benthem. *Logical Dynamics of Information and Interaction*. Cambridge University Press, 2011.