PriCL: Creating a Precedent, a Framework for Reasoning about Privacy Case Law

Michael Backes, Fabian Bendun, Jörg Hoffmann, and Ninja Marnau

CISPA, Saarland University
{backes,bendun,hoffmann,marnau}@cs.uni-saarland.de

Abstract. We introduce PriCL: the first framework for expressing and automatically reasoning about privacy case law by means of precedent. PriCL is parametric in an underlying logic for expressing world properties, and provides support for court decisions, their justification, the circumstances in which the justification applies as well as court hierarchies. Moreover, the framework offers a tight connection between privacy case law and the notion of norms that underlies existing rule-based privacy research. In terms of automation, we identify the major reasoning tasks for privacy cases such as deducing legal permissions or extracting norms. For solving these tasks, we provide generic algorithms that have particularly efficient realizations within an expressive underlying logic. Finally, we derive a definition of deducibility based on legal concepts and subsequently propose an equivalent characterization in terms of logic satisfiability.

1 Introduction

Privacy regulations such as HIPAA, COPPA, or GLBA in the United States impose legal grounds for privacy [25,30,31]. In order to effectively reason about such regulations, e.g., for checking compliance, it is instrumental to come up with suitable formalizations of such frameworks along with the corresponding automated reasoning tasks.

There are currently two orthogonal approaches to how regulations are expressed and interpreted in real life that both call for such a formalization and corresponding reasoning support. One approach is based on explicit rules that define what is allowed and what is forbidden. The alternative is to consider precedents, which is the approach predominantly followed in many countries such as the US. Precedents are cases that decide a specific legal context for the first time and thus serve as a point of reference whenever a future similar case needs to be decided. Moreover, even judges in countries that do not base their legal system on precedents often use this mechanism to validate their decision or shorten the process of argumentation.

Case law is particularly suitable for resolving vague formulations that naturally occur in privacy regulations like the definition of 'disclosure' in COPPA. Here, case law could reference decisions that define what circumstances are qualified as a non-identifiable form of personal data, thereby aiding the user by providing judicially accurate interpretation of such terms.

R. Focardi and A. Myers (Eds.): POST 2015, LNCS 9036, pp. 344–363, 2015.
DOI: 10.1007/978-3-662-46666-7_18

While rule-based frameworks have received tremendous attention in previous research (see the section on related work below) there is currently no formalization for case law that is amenable to automated reasoning.

Our Contribution. Our contribution to this problem space is threefold:

- We derive important legal concepts from actual judicial processes and relevant requirements from related work. The resulting framework PriCL, can be applied to the judicature of many different countries as it does not assume any specific argumentation.
- We tailor the framework for privacy regulations. In particular, our privacy specific case law framework is compatible with former policy languages since it has only minimal requirements regarding the logic. Therefore, it is possible to embed other formalizations into our framework.
- We define the major reasoning tasks that are needed to apply the framework to privacy cases. In particular, these tasks allow us to derive requirements for the underlying logic which we analyze. Several logics allow an embedding of the reasoning tasks by giving an equivalent characterization of the tasks. Consequently, we are able to select a well suited logic.

Related Work. There are plenty of privacy regulations that companies are required to comply with. In the US there are regulations for specific sectors, e.g., HIPAA for health data, COPPA for children's data, or GLBA and RFPA for financial data. In the EU, the member states have general data protection codes. The legislative efforts to harmonize these national codes via the EU Data Protection Regulation [18] are proceeding and already provide for identifying legislative trends. The importance and impact of these privacy regulations has brought the interpretation thereof to the attention of more technically focused privacy research [22,8,2,17,13,26].

Policy languages were mainly developed in order to model these regulations and to reflect companies' policies. Many of the modern logics modeling regulations are based on temporal logic [19,10,15,29,9] and were successfully used to model HIPAA and GLBA [16]. While these logics focus on expressiveness in order to reflect the regulations, the logics for company policies focus on enforcement [7,3] and thus also on authorization [1,3]. Consequently, company policies are mostly based on access control policies [24,21].

Bridging the gap between the regulation policies and the company's policies leads to automating compliance checks [28]. For many deployed policies, i.e., the ones that are efficiently enforceable, this is currently not possible due to the lack of decidability regarding the logics used to formalize regulations. However, for these cases there exist run-time monitoring tools that allow compliance auditing on log files [8,19,11,10]. In particular, such auditing was invented for HIPAA [19].

A different approach for achieving compliance is guaranteeing privacy-by-design [23,14,20]. However, the policy of these systems still needs to be checked for compliance with the relevant privacy regulations.

2 Ingredients

In the first step we illustrate which components are essential for a case law framework. To that end, we analyze actual judicial processes and derive ingredients for the framework from the relevant legal principles. Hence, in the following, we analyze a representative court decision[1] and discuss the implications for our framework.

The Conflict. *"This matter involves three certified questions from the Circuit Court of Harrison County regarding whether applicable state and federal privacy laws allow dissemination of confidential customer information [...] during the adjustment or litigation of an insurance claim."*

Every case reaching a court is based on a conflict, i.e., there is some question, as the one above, for which different parties have different opinions on its truth value. As a requirement for the framework, we can conclude that there has to be a conflict that needs to be resolved by a decision. This decision can be an arbitrary statement; hence, we call it a *decision formula*.

Sub-cases. A decision's justification usually involves decisions of several *sub-cases* in order to arrive at the final decision formula, e.g. the court needs to decide whether a specific law is applicable before examining what follows from its application. Each of these individual sub-case decisions may become a precedent for decisions which deal with a similar sub-case.

The Circumstances. *"[The plaintiff] concedes that under the definitions of the GLBA [...] information he requests is technically nonpublic personal information of a customer which the Act generally protects from disclosure[...]."*

Every case contains some factual background. These facts constitute some statements which are not under discussion but measurably true, e.g., that an address is nonpublic personal information. We summarize these facts in a *case description*.

Referencing Related Court Decisions. *"[T]he United States District Court for the Southern District of West Virginia handed down an opinion in Marks v. Global Mortgage Group, Inc., 218 F.R.D. 492 (S.D.W.Va.2003), providing us with timely and pertinent considerations."*

The key of case law is referencing other cases in order to derive statements. In the example case, this capability is used to introduce an argumentation from a different court. This mechanism is also used when statements are derived from regulations. Consequently, the framework has to be capable of introducing statements during the case justification by *references* to their origin.

Argumentation Structure of the Justification. *"[The] GLBA provides exceptions to its notification and opt-out procedures, including [...]"*

[1] The quotes are taken from MARTINO v. BARNETT, Supreme Court of Appeals of West Virginia, No. 31270, Decided: March 15, 2004. The decision text is public at `http://caselaw.findlaw.com/wv-supreme-court-of-appeals/1016919.html`

The argumentation structure of the justification is not linear, i.e., of the form $A \Rightarrow B \Rightarrow \ldots \Rightarrow$. But the arguments can be ordered in a tree form. The exceptions stipulated by the GLBA are enumerated and then discussed in the case justification. If more than one is applicable, these may serve as *independent decision grounds*, each being a potential precedent in its own right.

World Knowledge. *"[We] conclude that nonpublic personal information may be subject to release pursuant to judicial process."*

In the argumentation, the court leaves to the reader's knowledge that the plaintiff's litigation actually is a "judicial process". These open ends in the argumentation are neither explicitly covered by a decision nor by a case reference. Therefore, we need some world knowledge KB_W that will cover these axiomatic parts of the argumentation.

Precedents and Stare Decisis. The doctrine of *stare decisis* (to stand by things decided) or binding precedents is unique to common law systems. The decisions of superior courts are binding for later decisions of inferior courts (*vertical stare decisis*). These binding precedents are applied to similar cases by analogy.

In addition to the binding precedent, there also exists the persuasive precedent: *"While we recognize that the decision of the Marks court does not bind us, we find the reasoning in Marks regarding a judicial process exception to the GLBA very persuasive and compelling".*

Stare decisis does not apply in civil law systems, like those of Germany or France. However, these systems have a *jurisprudence constante*, facilitating predictable and cohesive court decisions. Though civil law judges are not obliged to follow precedents, they may use prior decisions as persuasive precedents and oftentimes do so.

Material Difference. Stare decisis only applies if the subsequent court has to decide on a case or sub-case that is similar to the precedent. Therefore, if the court finds *material difference* between the cases, it is not bound by stare decisis. In practice, judges may claim material difference on unwarranted grounds, which may lead to conflicting decisions of analoguous cases within our framework. Thus, we need to be able to account for *false material difference.*

Involving Court Hierarchies. *"[W]e look initially to federal decisions interpreting the relevant provisions of the GLBA for guidance with regard to the reformulated question. However, the issue proves to be a novel one in the country since few courts, federal or state, have addressed the exceptions to the GLBA."*

For our framework we need to take into account court hierarchies to identify binding precedents. In common law jurisdictions, inferior courts are bound by the decisions of superior courts; in civil law jurisdictions superior courts usually have higher authority without being strictly binding. In federal states like the USA or Germany we need to account for parallel hierarchies on state and on federal levels. This complex hierarchy has significant implications on stare decisis.

Hence, in our framework every case needs to be annotated by a court which is part of a *court hierarchy*, to identify the character of precedents, binding or potentially persuasive.

Ratio Decidendi and Obiter Dicta. Regarding the court's decision text, we need to differentiate between two types of statements. The actual binding property of a precedent has only those statements and legal reasoning that are necessary for the rationale of the decision. These necessary statements as called *ratio decidendi* and constitute the binding precedent. Further statements and reasoning that are not essentially necessary for the decision are called *obiter dicta*. These are not binding but can be referenced as persuasive precedents.

For our reasoning framework we need to differentiate and annotate statements into these two different categories to correctly identify binding precedents.

3 Defining the PriCL Framework

Reflecting the observations just made, we define cases (Section 3.1) and case law databases (Section 3.2). Thereby we also explain how to model the legal principles described in Section 2. Then, we define how the database can be used in order to deduce facts outside the framework (Section 3.3). We analyze our framework, validating a number of basic desirable properties of case law databases (Section 3.4). We finally show, for privacy regulations specifically, that our framework matches the requirements identified by previous work [8] (Section 3.5).

Throughout this section, we assume an underlying logic in which world properties are expressed and reasoned about. Our framework is parametric with respect to the precise form of that logic. The requirements the logic has to fulfill are interpreting predicates as relations over objects, supporting universal truth/falseness (denoted respectively as $\top$ and $\bot$), conjunction (denoted $\wedge$), entailment (denoted $A \models B$ if formula A entails formula B), and monotonicity regarding entailment, i.e., if $A \models B$ then $A \wedge C \models B$ for any formula C. As an intuition when reading the following, the reader may assume we are using a first-order predicate logic.

3.1 Introducing Cases

As we have seen, a case consists of a decision formula, a case description, a court, and a proof tree. The first three components are straightforward to capture formally (courts are represented by a finite set Courts of court identifiers). Designing the proof tree is more involved since it needs to capture the judge's justification. We distinguish between different kinds of nodes in the tree depending on the role the respective statements play in the justification: Does a sentence make an axiomatic statement, or form part of the case description? Does it refer to a previous case, adopting a decision under particular prerequisites? Does it make an assessment on the truth of a particular statement (e.g., that a particular piece of information is or is not to be considered private) under particular prerequisites?

We therefore reflect these "standalone" statements in the leaf nodes of the proof tree, categorized by the three different types of statements mentioned.

The inner nodes of the tree perform logical deductions from their children nodes, representing the reasoning inherent in the justification, i.e., the conclusions that are made until finally, in the tree root, the decision formula is reached. We differentiate between two kinds of reasoning steps, AND-steps and OR-steps.The OR-steps reflect the principle of *independent decision grounds*. The AND-step is the natural conclusion steps that is used to ensure that the decision made is reached through the argumentation.

In order to avoid a recursive definition, we need a (possibly infinite) set of case identifiers $\mathcal{C}_I$. Throughout the paper we assume a fixed given set $\mathcal{C}_I$.

Definition 1 (Case). *A case C is a tuple* $(\mathsf{df}, \mathsf{CaseDesc}, \mathsf{ProofTree}, \mathsf{crt})$ *s.t.*

- df *is a formula that we call the* decision formula *of* C.
- $\mathsf{CaseDesc}$ *is a formula describing the case's circumstances.*
- $\mathsf{ProofTree}$ *is a (finite) tree consisting of formulas* f *where the formula of the root node is* df. *Inner nodes are annotated with* AND *or* OR *and leaves are annotated with* $l \in \{\mathsf{Axiom}, \mathsf{Assess}\} \cup \{\mathsf{Ref}(i) \mid i \in \mathcal{C}_I\}$. *Leaf formulas* l *are additionally associated with a* prerequisite *formula* pre. *For leaves annotated with* Axiom, *we require that* $\mathsf{pre} = l$.
- $\mathsf{crt} \in \mathsf{Courts}$.

For leaf formulas l, *we refer to* l *as the node's* fact, *and we will often write these nodes as* $\mathsf{pre} \to \mathsf{fact}$ *where* $\mathsf{fact} = l$.

By the prerequisites *of an inner node* n *with children nodes* $n_1, \ldots, n_k$, *denoted as* $\mathsf{pres}(n)$, *we refer to* $\bigvee_{1 \leq i \leq k} \mathsf{pres}(n_i)$ *if* n *is annotated by* OR *and* $\bigwedge_{1 \leq i \leq k} \mathsf{pres}(n_i)$ *if* n *is annotated by* AND. *The prerequisites of a case* C *are the prerequisites of the root node and denoted by* pres_C. *We define analogously the* facts *of a node and a case. We will often identify formulas with proof tree nodes. Given a case* C, *by* df_C *we denote the decision formula of* C.

Let $\mathbf{C}$ *be a set of cases and* $\mu : \mathbf{C} \to \mathcal{C}_\mathbf{I}$ *a function. If for every reference* $\mathsf{Ref}(i)$ *in* $\mathbf{C}$, *there is an* $D \in \mathbf{C}$ *with* $\mu(D) = i$, *we call the set* $\mathbf{C}$ closed *under* μ.

We assume *world knowledge* common to all cases. In the example of argumentation ends in Section 2, it is assumed that the reader knows that the predicate $\mathsf{is\_judical\_process}$ holds for any case. Formally, the world knowledge is a formula KB_W (naturally, a conjunction of world properties) in the underlying logic.

Definition 1 is purely syntactic, imposing no restrictions on how the different elements are intended to behave. We will fill in these restrictions one by one as part of spelling out the details of our framework, forcing cases to actually decide a conflict and behave according to the legal principles. One thing the reader should keep in mind is that $\mathsf{pre} \to \mathsf{fact}$ is *not* intended as a logical implication. Rather, pre are the prerequisites that a judge took into account when making the assessment that fact (e.g., the privacy status of a piece of information) is considered to be true under the circumstances $\mathsf{CaseDesc} \models \mathsf{pre}$. This solely captures human decisions such as trade-off decisions. However, the frameworks allows reasoning about consequence of such decisions. The formulas pres_C, and

respectively facts_C, collect all prerequisites needed to apply the proof tree, and respectively all facts needed to execute the proof tree; axiom leaves act in both roles.

In principle, a case has the purpose to decide a formula df. However, while justifying that a formula holds, e.g., that a telecommunication company has to delete connection data after a certain amount of time, the court might decide other essential subquestions. This concept is conveniently captured through the notion of *subcases*.

Definition 2 (Subcase). *Let* $C = (\mathsf{df}, \mathsf{CaseDesc}, \mathsf{ProofTree}, \mathsf{crt})$ *be a case and* $n \in \mathsf{ProofTree}$ *a node. Let* $\mathsf{sub}(n)$ *be the subtree of* $\mathsf{ProofTree}$ *with root node* n*. The case* $\mathsf{sub}(C, n) := (n, \mathsf{CaseDesc}, \mathsf{sub}(n), \mathsf{crt})$ *is a* subcase *of* C*.*

Another aspect that is of interest when referencing cases is the degree of abstraction. For example, one case could decide that a specific telecommunication company C has to delete connection information D of some user U after a specific time period t. The question of how this decision can be used in order to decide the question for different companies C' or different information D' is covered by the legal concept of material difference. For this work, we assume that a judge specifies the allowed difference in the prerequisites of a decision.

Our definition of cases, so far, is generic in the sense that it may be applied to any domain of law. To configure our framework to privacy regulations more specifically, a natural approach is to simply restrict the permissible forms of decision formulas. We explicitly leave out legal domains such as individualized sentencing or measuring of damages. Decisions in the privacy context are about whether or not a particular action is legal when executed on particular data. We capture this by assuming a dedicated predicate $\mathsf{is\_legal\_action}$, and restricting the decision formula to be an atomic predicate of the form $\mathsf{is\_legal\_action}(a)$, where a is an action from an underlying set $\mathsf{Actions}$ of possible actions treated as objects (constants) in the underlying logic. This can also be used in other legal domains, but it turns out to be sufficient to connect our formalization of privacy cases with other policy based approaches. Note that, in contrast to other policy frameworks, we do not need to add the context to the predicate, as the context is contained in the case, via nodes of the form *"if the transfer-action* a *has purpose marketing and the receiver is a third party, then* $\neg\mathsf{is\_legal\_action}(a)$*"*. As decisions about the legality of actions are not naturally part of the common world knowledge KB_W, nor of the case description $\mathsf{CaseDesc}$ itself, our modeling decision is to disallow the use of $\mathsf{is\_legal\_action}$ predicates in these formulas. In other words, the world and case context describe the circumstances which are relevant to determining action legality, but they do not define whether or not an action is legal.

Definition 3 (Privacy Case). *Given world knowledge* KB_W *and action set* $\mathsf{Actions}$*, a case* $C = (\mathsf{df}, \mathsf{CaseDesc}, \mathsf{ProofTree}, \mathsf{crt})$ *is a* privacy case *if* $\mathsf{df} \in \{\neg\mathsf{is\_legal\_action}(a), \mathsf{is\_legal\_action}(a)\}$ *for some action* $a \in \mathsf{Actions}$*, where the* $\mathsf{is\_legal\_action}$ *predicate is not used in either of* KB_W *or* $\mathsf{CaseDesc}$*.*

Starting to fill in the intended semantics of cases, we first capture the essential properties a case needs to have to "make sense" as a stand-alone structure. Additional properties regarding cross-case structures will be considered in the next subsection. We will use the word "consistency" to denote this kind of property. The following definition captures the intentions behind cases:

Definition 4 (Case Consistency). *Let* $C = (\mathit{df}, \mathit{CaseDesc}, \mathit{ProofTree}, \mathit{crt})$ *be a case.* C *is* consistent *if the following holds (for all nodes* n *where* $n_1, \ldots, n_k$ *are its child nodes)*

(i) $\mathit{KB}_W \wedge \mathit{CaseDesc} \not\models \bot$ *(ii)* $\mathit{KB}_W \wedge \mathit{CaseDesc} \models \mathit{pres}_C$

(iii) $\mathit{KB}_W \wedge \mathit{CaseDesc} \wedge \mathit{facts}_C \not\models \bot$

(iv) $\bigwedge_{1 \leq i \leq k} n_i \models n$ *if* n *is an* AND *step and* $\bigvee_{1 \leq i \leq k} n_i \models n$ *if* n *is an* OR *step*

Regarding (i), if the world knowledge contradicts the case description, i.e., $\mathsf{KB}_W \wedge \mathsf{CaseDesc} \models \bot$, then the case could not have happened. Similarly, (iii) the case context must not contradict the facts that the proof tree makes use of (this subsumes (i), which we kept as it improves readability). As for (ii), the case context must imply the axioms as well as the prerequisites which the present judge (assessments) or other judges (references to other cases; see also Definition 7) assumed to conclude these facts. (iv) says that inner nodes must represent conclusions drawn from their children.

The OR nodes of the proof tree reflect the legal argumentation structure of *independent decision grounds*, the judge gives several arguments. If the judge of a later case decides that one of these arguments is invalid for the conclusion, he needs to be able to falsify only one of the branches and not the whole tree.

3.2 Combining Cases to Case Law Databases

The quintessential property of case law is that cases make references to other cases. These references are necessary to formulate several legal principles.

The legal principles *false material difference* and *reversing decisions* define requirements for when not to reference a case, either because it contains a mistake or because the opinion has changed over time. Therefore, we consider the design cleaner if both principles are covered by the same mechanism of the framework and hence we denote single Assess nodes as unwarranted, i.e., to forbid the reference to be used thereafter.

We require a different mechanism to differentiate cases we must agree with and cases which we may use as reference. Unwarranting rather defines which decisions must not be referenced. In particular, we need to differentiate between assessments coming from the legal principles *ratio decidendi* and *obiter dicta*. While the part of the decision following *ratio decidendi* leads to a binding precedent, the *obiter dicta* part is not binding. Thus, we introduce predicates may-ref and must-agree. It also provides a mechanisms to respect the *court hierarchy*. Intuitively, $\mathsf{may\text{-}ref}(C_1, C_2)$ denotes the circumstances that case C_1 may reference case C_2; $\mathsf{must\text{-}agree}(C_1, C_2)$ analogously denotes that C_1 must agree with C_2.

In addition, we need to introduce the concept of time by a total order $\leq_t$ over cases. This concept allows us to formulate the requirement that references can only point to the past.

Definition 5 (Case Law Database (CLD)). *A* case law database *is a tuple* $\mathsf{DB} = (\mathbf{C}, \leq_t, \mathsf{must\text{-}agree}, \mathsf{may\text{-}ref}, \mu, U)$ *such that:*

- $\mathbf{C}$ *is a set of cases. We will also write* $C \in \mathsf{DB}$ *for* $C \in \mathbf{C}$.
- $\mu : \mathbf{C} \to \mathcal{C}_\mathbf{I}$ *is an injective function such that* $\mathbf{C}$ *is closed under* μ. *In the following we will also write* $\mathsf{Ref}(D)$ *for* $\mathsf{Ref}(i)$ *if* $\mu(D) = i$.
- *Let* $<_{\mathsf{ref}} := \{(C, D) \mid D \text{ contains a } \mathsf{Ref}(C) \text{ node}\}$ *and* $\leq_t$ *is an order that we call* time order *of the cases. It has to hold:*
$$\mathsf{must\text{-}agree} \subseteq <_{\mathsf{ref}} \subseteq \mathsf{may\text{-}ref} \subseteq \leq_t \subseteq \mathbf{C} \times \mathbf{C}$$
- U *specifies the unwarranted nodes, i.e.,* $U : \mathbf{C} \to \mathbf{N}$ *is function such that*
  - $\mathbf{N}$ *is a subset of the nodes labelled with* Assess *or* Ref *in the cases* $\mathbf{C}$.
  - *The set increases monotonic, i.e.,* $C \leq_t D \implies U(C) \subseteq U(D)$.

  We denote the unwarranted *nodes of* DB *by* $U(\mathsf{DB}) := \bigcup_{C \in \mathbf{C}} U(C)$.

The function μ is used to remove the recursive definition of a case and enables us to connect cases via their individual semantics.

Regarding the relations must-agree and the may-ref we made two design decisions. First, we require to not link must-agree and the actual references $<_{\mathsf{ref}}$. On the one hand, there might be precedents which are not applicable, but on the other hand, we want the freedom to define must-agree and may-ref only depending on the court hierarchy. The second design decision is to base these relations on cases instead of decision nodes. As for the first decision, the purpose is to make an instantiation of the definition only depending on the court, but we need to be careful regarding the principles *ratio decidendi* and *obiter dicta*. Since one of them is not binding, i.e., a must-agree and the other is. This differentiation can be achieved by replacing every case with a set of cases. We require this to be part of the modeling process. We did not add further restrictions since they may depend on local law.

Example 1 (Must-agree and may-references for a court hierarchy). Assume the set of courts Courts is partially ordered by $\leq_\S$, i.e., there is a court hierarchy. In this case, we could model must-agree by
$$\mathsf{must\text{-}agree} = \{(C_1, C_2) \mid C_i = (\mathsf{df}_i, d_i, p_i, \mathsf{crt}_i), i \in \{1, 2\}, C_1 \leq_t C_2, \text{ and } \mathsf{crt}_1 \leq_\S \mathsf{crt}_2\}.$$
It is easy to see that the must-agree predicate actually only depends on the crt and not on the other parameters of the proof. We call this property *court-dependency*.

The key property of unwarranted decisions is that they are time dependent. In order to only use warranted decisions when referencing, we define warranted subcases as follows:

Definition 6 (Warranted Subcase). *A subcase* $(\mathit{df}, \mathit{CaseDesc}, \mathit{ProofTree}, \mathit{crt})$ *is* warranted *with respect to a set* N *of nodes if the case* $(\mathit{df}, \mathit{CaseDesc}, \mathit{ProofTree}', \mathit{crt})$ *is consistent where* $\mathit{ProofTree}'$ *is derived from* $\mathit{ProofTree}$ *by replacing every precondition of a node* $n \in N$ *by* $\bot$.

It remains to define when a case law database can be considered to be consistent. To that end, we consider case references and conflicts between cases. Starting with the former, we obtain:

Definition 7 (Correct Case Reference). *Let* DB *be a case law database and* $C = (\mathit{df}, \mathit{CaseDesc}, \mathit{ProofTree}, \mathit{crt})$ *a case in* DB. *A leaf node* $\mathsf{pre} \rightarrow \mathsf{fact}$ *in* $\mathit{ProofTree}$ *annoted with* $\mathit{Ref}(D)$ references correctly *if* $D_u = (\mathsf{fact}, \mathit{CaseDesc}_D, \mathit{ProofTree}_D, \mathit{crt}_D)$ *is a warranted subcase of a case* $D \in \mathit{DB}$ *w.r.t.* $U(C)$, $\mathit{may\text{-}ref}(C, D)$ *holds and* $\mathsf{KB}_W \wedge \mathsf{pre} \models \mathsf{pres}_D$. C references correctly *if all its leaves annoted with* $\mathit{Ref}(D)$ *reference correctly.*

Consider that, when referencing a (sub)case D as $\mathsf{pre} \rightarrow \mathsf{fact}$ from our case C at hand, we are essentially saying that the same argumentation applied in D can be applied in our case, to prove fact under circumstances pre. So we need to show that this applicability of arguments is actually given. This is ensured by $\mathsf{KB}_W \wedge \mathsf{pre} \models \mathsf{pres}_D$ because pres_D collects all prerequisites, axioms and otherwise, needed to apply D. Note that, if C is consistent, by Definition 4 (ii) it holds that $\mathsf{KB}_W \wedge \mathsf{CaseDesc} \models \mathsf{pre}$ and thus $\mathsf{KB}_W \wedge \mathsf{CaseDesc} \models \mathsf{pres}_D$. As the same applies recursively to the case references made in D, we know that pre (given KB_W and $\mathsf{CaseDesc}$) entails *all* judge decisions underlying the assessment fact.

We are now almost in the position to define consistency of the entire case law database. The last missing piece in the puzzle is to identify when cases should be considered to be in conflict — which naturally occurs in case law databases where judges may make different decisions. We capture this through pairs of cases whose prerequisites are compatible, while their facts are contradictory:

Definition 8 (Case Conflict). *Let* C_1 *be a case in* DB *and* C_2 *be a warranted case w.r.t.* $U(C_1)$. *We say that* C_1 *is* in conflict with C_2 *if and only if*

(i) $\mathsf{KB}_W \wedge \mathsf{pres}_{C_1} \wedge \mathsf{pres}_{C_2} \not\models \bot$ *(ii)* $\mathsf{KB}_W \wedge \mathsf{facts}_{C_1} \wedge \mathsf{facts}_{C_2} \models \bot$

(iii) $\mathit{must\text{-}agree}(C_1, C_2)$

A case C *is in conflict with* DB *if there is a* $D \in \mathit{DB}$ *s.t.* C *is in conflict with* D.

We ignore the case descriptions here, other than what is explicitly employed as axioms in the proof trees: we consider cases to be in conflict if one *could* construct a case (e.g., $\mathsf{pres}_{C_1} \wedge \mathsf{pres}_{C_2}$) which would make it possible to come to a contradictory decision. We define case law database consistency as follows:

Definition 9 (Case law database consistency). *A case law database* $\mathit{DB} = (\mathbf{C}, \leq_t, \mathit{must\text{-}agree}, \mathit{may\text{-}ref}, \mu, U)$ *is*

(i) case-wise consistent *if every* $C \in \mathit{DB}$ *is consistent,*

(ii) referentially consistent *if every* $C \in \mathit{DB}$ *references correctly, and*

(iii) hierarchically consistent *if every* $C \in \mathit{DB}$ *is not in conflict with* DB.

(iv) warrants consistently *if for every C holds: $U(C)$ contains all $\mathsf{Ref}(D)$ nodes where D is an unwarranted subcase w.r.t. $U(C)$.*

We call DB consistent *if it warrants consistently and is hierarchically, referentially and case-wise consistent.*

3.3 Deriving Legal Consequences: Deducibility and Permissibility

In the following we assume that the predicates may-ref and must-agree of the DB do not depend on the case description, the decision formula or the proof tree, but are only court dependent, cf. Example 1. As a consequence, we know the value of these predicates for formula values and case descriptions which are not contained as a case in the database given only the court level of the case. In other words, we require an operation $\mathsf{DB} \cup \{C\}$ that puts C at the end of the timeline regarding $\leq_t$, assigns a fresh identifier $i \in \mathcal{C}_I$ to C with μ, uses as $U(C) := U(\mathsf{DB})$, and adopts must-agree, may-ref appropriately and is independent of the decision formula and the proof tree. This operation is needed to apply the framework to situations not contained in the database.

Obvious applications of our framework are advanced support for case search, and consistency checking. A more advanced task is to evaluate the legality of actions given the cases reflected in the database. For example, when designing a course administration system, one may ask "Am I allowed to store students' grades in the system?" Our formalism supports this kind of question at different levels of strength, namely:

Definition 10 (Deducibility and Permissibility). *Let $DB = (\mathbf{C}, \leq_t, \mathsf{must\text{-}agree}, \mathsf{may\text{-}ref}, \mu, U)$ be a consistent CLD, and f a formula. We say that f is* permitted *in DB under circumstances CaseDesc and court crt if there exists a case $C = (f, \mathsf{CaseDesc}, \mathsf{ProofTree}, \mathsf{crt})$ such that ProofTree does not contain nodes labeled with Assess, and $DB \cup \{C\}$ is consistent (where C is inserted at the end of the timeline $\leq_t$). We say that f is* uncontradicted *in DB under CaseDesc and crt if $\neg f$ is not permitted under CaseDesc and crt. We say that f is* deducible *if it is permitted and uncontradicted.*

For sets F of formulas, we say that F is permitted in DB under CaseDesc and crt if there exists a set of cases $\{C_f = (f, \mathsf{CaseDesc}, \mathsf{ProofTree}_f, \mathsf{crt}) \mid f \in F\}$ such that every $\mathsf{ProofTree}_f$ does not contain nodes labeled with Assess, and $DB \cup \{C_f \mid f \in F\}$ is consistent (where the C_f are inserted in any order at the end of the timeline $\leq_t$).

It might be confusing at first why we attach to f the weak attribute of being "permitted" if we can construct a case supporting it. The issue is, both f and $\neg f$ may have such support in the same database. This follows directly from the freedom of different courts to contradict each other. If two courts at the same level decide differently on the same issue, then that is fine by our assumptions. Hence, to qualify a formula f for the strong attribute of being "deducible", we require the database to permit f and to not permit its contradiction.

The concept of deducibility of a *set* F of formulas is interesting because, in general, this is not the same as deducing each formula in separation. In particular, while each of f and $\neg f$ may be permitted in the same database, $\{f, \neg f\}$ is never permitted because adding the hypothetical supporting cases necessarily incurs a hierarchical conflict. Permissibility of F is also not the same as permissibility of $\bigwedge_{f \in F} f$ because the latter makes a stronger assumption: all cases referred to in order to conclude $\bigwedge_{f \in F} f$ must have compatible prerequisites. So deducibility of formula sets forms a middle ground between individual and conjunctive deducibility.

Theorem 1. *There is a consistent case law database* DB*, case description* $\mathsf{CaseDesc}$ *and court* crt*, such that there is a set* F *of formulas for each of the following properties (in* DB *under circumstances* $\mathsf{CaseDesc}$ *and court* crt*):*

(i) For every $f \in F$*,* f *is permissible and* F *is not permissible.*
(ii) F *is permissible, but* $\bigwedge_{f \in F} f$ *is not permissible.*

The proof and the details of all other proofs are given in the long version [6].

Characterizing Deducibility. Deducibility is the central concept for answering questions that are not explicitly answered by the database. However, Definition 10 does not give an algorithmic description of how to decide whether some formula is deducible. It is also inconvenient for proving properties about permissibility and deducibility.

Intuitively, a formula should be permissible if there is a set of warranted decisions which allow us to conclude the predicate and a formula f should be deducible if in addition no set of decisions contradicts f. We will first define *supporting sets* and then prove that the intuition matches the definitions of permissibility and deducibility.

Definition 11 (Supporting set). *Let* $\mathsf{DB} = (\mathbf{C}, \leq_t, \mathsf{must\text{-}agree}, \mathsf{may\text{-}ref}, \mu, U)$ *be a consistent case law database,* f *a formula,* $\mathsf{CaseDesc}$ *a case description and* crt *a court. A set* $\mathcal{A}$ *of leaf nodes in* DB *that are labeled with* Assess *is a* supporting set *for formula* f *if the following holds:*

(1) $\mathsf{KB}_W \wedge \mathsf{CaseDesc} \models \bigwedge_{(\mathsf{pre} \to \mathsf{fact}) \in \mathcal{A}} \mathsf{pre}$
(2) $\mathsf{KB}_W \wedge \mathsf{CaseDesc} \wedge \bigwedge_{(\mathsf{pre} \to \mathsf{fact}) \in \mathcal{A}} \mathsf{fact} \models f$
(3) $\mathsf{KB}_W \wedge \mathsf{CaseDesc} \wedge \bigwedge_{(\mathsf{pre} \to \mathsf{fact}) \in \mathcal{A}} \mathsf{fact} \not\models \bot$

A supporting set is unwarranted if it contains an unwarranted node w.r.t. any $C \in \mathbf{C}$*. If it is not unwarranted it is warranted.*

A supporting set is consistent *with* DB *if* $\mathsf{DB} \cup \{(\top, \mathsf{CaseDesc}, \mathsf{ProofTree}, \mathsf{crt})\}$ *is consistent, where* $\mathsf{ProofTree}$ *consists of a root node with annotation* $\top$ *and leaf nodes with annotation* $\mathsf{Ref}(C_n)$ *for* $n \in \mathcal{A}$*, where* C_n *is the case that contains node* n*.*

Note that a supporting set that is consistent with the DB leads to consistency, and correct referencing, and does not create any conflicts. The properties

required in the definition are a consequence of the definition of database consistency. A case constructed from a supporting set would simply refer to all decisions and place the formula at the root.

The following theorem characterizes permissibility and deducibility using supporting sets. This characterization suggests an algorithmic way of deciding the properties and gives a tool for proving properties about case law databases.

Theorem 2. *Let DB be a consistent case law database, f a formula, CaseDesc a case description and crt a court. The following holds:*

1. *$C \in DB$ with warranted node $f \Rightarrow \exists \mathcal{A}$ that supports f*
2. *f is permitted (under circumstance CaseDesc and court crt) $\Leftrightarrow \exists \mathcal{A}$ that supports f, is warranted, and is consistent with DB*
3. *f is deducible $\Leftrightarrow \exists \mathcal{A}$ that supports f and is consistent with DB, and $\forall \mathcal{B}$ it holds that $\mathcal{B}$ does not support $\neg f$, is unwarranted, or is not consistent with DB*

3.4 General Properties of Case Law Databases

Introducing a new framework always comes with the risk of modeling errors. A method for alleviating that risk is to prove properties that the framework is expected to have. In order to validate the framework introduced here, we have proven that (i) case references do not influence decisions (Theorem 2); in this subsection we also prove that (ii) consistency is necessary for property (i) (Theorem 3), and that (iii) neither $\bot$ nor $\{f, \neg f\}$ are ever permitted (Theorem 4).

Regarding (i), we have shown that every formula f in the database can be derived from a supporting set of previous decisions (Theorem 2) with the case description and world knowledge. Hence there is no possible interplay between case references that would make it possible to prove something not backed up by judges' decisions.

Regarding (ii), Theorem 2 implies immediately that, whenever a formula f is deducible, then it follows from decisions made by judges in previous cases. It is easy to verify that our restrictions are necessary to ensure this, i.e., that this property gets lost if we forsake either case-wise or referential consistency:

Theorem 3. *Let DB be a case law database, and let f be any formula that does not entail $\bot$. Then there exist cases C_1 and C_2, each with root node f and the empty case desc $\top$, such that (inserting C_i at the end of the timeline $\leq_t$):*

- *If DB is case-wise consistent, then so is $DB \cup \{C_1\}$.*
- *If DB is referentially consistent, then so is $DB \cup \{C_2\}$.*
- *If there is a crt such that must-agree(crt) $= \emptyset$, then in addition this holds: for each of $i = 1, 2$, if DB is hierarchically consistent, then so is $DB \cup \{C_i\}$.*

We remark that, by restricting the formula f only slightly, the proof of Theorem 3 can be strengthened so as not to have to rely on a maximal court for ensuring hierarchical consistency. In particular, if f is made of predicates that do not occur anywhere in the case law database, then the cases C_1 and C_2 as constructed cannot be in conflict with any other cases, thus preserving hierarchical consistency for arbitrary courts crt. We finally prove (iii), non-permissibility of either $\bot$ or $\{f, \neg f\}$:

Theorem 4. *The formula $\bot$ is not permitted in any case law database DB, under any circumstances CaseDesc and court crt. The same holds for $\{f, \neg f\}$ if $crt \in$ must-agree(crt).*

3.5 Privacy Cases and Norms

We now point out an interesting property of privacy cases, and of databases consisting only of such cases. We call such databases *privacy case law databases*.

Rule based privacy policies are a well established and widely used concept. The rules that are used are usually reflected by norms defining privacy regulations. However, neither rules nor norms are reflected in the case law framework. In this subsection, we show that we can use a natural definition of norms that can be extracted from privacy cases. In addition, it is possible to transform a privacy case to a normal form such that a norm that decides the case is represented.

At the core of privacy regulations are positive and negative norms, as introduced by [8]. Positive norms are permissive in the sense that they describe conditions that allow transactions with personal data ($\phi \Rightarrow$ is_legal_action(a)). Negative norms, in contrast, define necessary conditions for such transactions, i.e., they forbid transactions with personal data unless certain conditions are met ($\phi \Rightarrow \neg$is_legal_action(a)).

Definition 12 (Norms). *Let $a \in$ Actions. A* norm *is a formula that has the form $\phi \Rightarrow p$ where is_legal_action(a) does not occur in ϕ. The norm is a positive norm, denoted ϕ^+, if $p =$ is_legal_action(a) and a negative norm, denoted ϕ^-, if $p = \neg$is_legal_action(a). A norm ϕ decides p given f if $KB_W \wedge f \models \phi$.*

In the case law framework, norms are hidden by judges' assessments. However, in the spirit of Theorem 2, norms are reflected by sets of cases that could be referenced in order to support either the legality of an action (positive norm) or its illegality (negative norm). In the following theorem, we show that we can extract a norm for every privacy case avoiding the recursion of Theorem 2.

Theorem 5. *Let DB be a consistent privacy case law database and $C = (df, CaseDesc, ProofTree, crt) \in DB$. Then there is a norm ϕ that decides df given CaseDesc. In particular, there are formulas ϕ_W, ϕ_S such that is_legal_action(a) does not occur in these formulas and (1) $facts_C \Rightarrow \phi_W \wedge (\phi_S \Rightarrow df)$ (2) $\phi_W \wedge (\phi_S \Rightarrow df) \Rightarrow df$*

The formulas ϕ_W and ϕ_S can be used to construct a normal form of privacy cases. In particular, this normal form is consistent and allows reading off norms.

Corollary 1 (Normal forms). *Let $DB = (\mathbf{C}, \leq_t, \text{must-agree}, \text{may-ref}, \mu, U)$ be a privacy case law database, $C = (df, CaseDesc, ProofTree, crt) \in DB$ be a case, and D be the set of C's leaf nodes. $N(C)$ is the case that consists of a root node df, two inner nodes ϕ_w and $\phi_S \Rightarrow df$ and the leaf nodes D as children of both inner nodes. We call $N(C)$ the* normal form *of C. If DB is consistent, then $(\mathbf{C} \backslash \{C\} \cup \{N(C)\}, \leq_t)$ is also consistent (where $N(C)$ is placed at the position of C w.r.t. $\leq_t$).*

In order to define $N(C)$, we need to duplicate the leaf nodes since the transformations to get ϕ_W and ϕ_S ignore which fact is needed to get the corresponding formula. Thus, a leaf node's fact could end up in both formulas ϕ_W and ϕ_S.

In conformance with [8], we can conclude from deducibility of an action that there is a positive norm supporting it and show that no negative norm can be applied, i.e., all negative norms are respected (Theorem 4).

4 Reasoning Tasks

We now discuss the reasoning tasks associated with our framework — how to answer questions such as "are we allowed to send data D to some party P?" — in more detail, giving an algorithm sketch and brief complexity analysis (in terms of the number of reasoning operations required) for each.

Consistency. Analyzing and keeping the state of the case law database consistent is of vital importance for its usefulness; cf. Theorem 4. As in the definition of consistency, we split the task of checking consistency into case-wise, referential, and hierarchical consistency. Due to their simplicity, we postpone the detailed description of their algorithms to the long version [6].

All of these properties are defined per case, i.e., the case wise check of the corresponding property has to be repeated $|\mathsf{DB}|$ times. Following the respective definition, checking case consistency costs $|\mathsf{ProofTree}+1|$ entailment operations and checking correct referencing for C costs $\mathsf{references}(C)$ where $\mathsf{references}(C)$ is the number of nodes in C annotated by $\mathsf{Ref}(D)$. Hierarchical consistency can be checked along the time line $\leq_t$ only testing for conflicts with earlier cases. So for the i-th case, we need at most $(i-1)\cdot 2$ entailment checks, since every conflict check requires 2. Consequently, we require $|\mathsf{DB}|\cdot(|\mathsf{DB}|+1)$ entailment checks.

Deducibility and Permissibility. As deducibility amounts to two consecutive permissibility checks, we consider the latter exclusively. We are given a database DB, a formula whose permissibility should be checked, as well as a case description $\mathsf{CaseDesc}$ and a court crt forming the circumstances.Permissibility is equivalent to the existence of a supporting set $\mathcal{A}$ for f that is consistent with DB. Thus the task of permissibility can be reduced to checking the existence of a suitable set $\mathcal{A}$. If the answer is "yes", we can also output a witness, i.e., a hypothetical case C showing permissibility. A straightforward means for doing this is to set $C := (f, \mathsf{CaseDesc}, \mathsf{ProofTree}, \mathsf{crt})$ where $\mathsf{ProofTree}$ consists of root node f, one leaf node l labeled with $\mathsf{Ref}(D)$ for every $D \in \mathcal{A}$, as well as one leaf node $\mathsf{KB}_W \wedge \mathsf{CaseDesc}$ labeled with Axiom. For convenience, we will denote this construction by $C(\mathcal{A})$. See Algorithm 1.

The correctness of the algorithm is shown by Theorem 2. In contrast to our previous algorithms, deducibility checking as per Algorithm 1 requires an exponential number of entailment checks in the worst case. This raises the questions (1) whether or not this exponential overhead is inherent in the complexity of deciding permissibility, and (2) whether it is possible to encode the permissibility test directly into the logic instead.

Algorithm 1. Permissibility

Input : A formula f, case description $\mathsf{CaseDesc}$, court crt, and a consistent CLD DB

Output: A case $C = (f, \mathsf{CaseDesc}, \mathsf{ProofTree}, \mathsf{crt})$ such that $\mathsf{DB} \cup \{C\}$ is consistent (where C is set to be the maximum w.r.t. $\leq_t$), or $\bot$ if no such C exists

1 Test whether $\mathsf{KB}_W \wedge \mathsf{CaseDesc} \models \bot$. If so, output $\bot$.
2 Test whether $\mathsf{KB}_W \wedge \mathsf{CaseDesc} \models f$. If so, output $(f, \mathsf{CaseDesc}, \mathsf{ProofTree}, \mathsf{crt})$ where $\mathsf{ProofTree}$ is the proof tree consisting of a leaf node labeled by Axiom containing f.
3 Set $\mathcal{N} := \emptyset$.
4 **for** *every* $D \in \mathsf{DB}$ *and every* $(\mathsf{pre} \to \mathsf{fact}) \in D$ *labeled* Assess **do**
5 | Check if $\mathsf{KB}_W \wedge \mathsf{CaseDesc} \models \mathsf{pre}$
6 | Check if $\mathsf{KB}_W \wedge \mathsf{CaseDesc} \wedge \mathsf{fact} \not\models \bot$
7 | If both checks succeed, set $\mathcal{N} := \mathcal{N} \cup \{(\mathsf{pre} \to \mathsf{fact})\}$.
8 **end**
9 **for** $\mathcal{A} \in 2^{\mathcal{N}}$ **do**
10 | Check that $\mathsf{KB}_W \wedge \mathsf{CaseDesc} \models \bigwedge_{(\mathsf{pre} \to \mathsf{fact}) \in \mathcal{A}} \mathsf{pre}$
11 | Check that $\mathsf{KB}_W \wedge \mathsf{CaseDesc} \wedge \bigwedge_{(\mathsf{pre} \to \mathsf{fact}) \in \mathcal{A}} \mathsf{fact} \models f$
12 | Check that $\mathsf{KB}_W \wedge \mathsf{CaseDesc} \wedge \bigwedge_{(\mathsf{pre} \to \mathsf{fact}) \in \mathcal{A}} \mathsf{fact} \not\models \bot$
13 | **for** *every* $E \in \mathsf{DB}$ *with* $\mathsf{crt} <_{\S} \mathsf{crt}_E$ **do**
14 | | Check that E and $C(\mathcal{A})$ are not in conflict.
15 | **end**
16 | If all three tests succeed, go on with step 18, otherwise continue with the next $\mathcal{D}$.
17 **end**
18 If a set $\mathcal{A}$ succeeded, output $C(\mathcal{A})$, otherwise output $\bot$.

The answer to (1) is a qualified "yes" in the sense that permissibility checking essentially pre-fixes entailment checks with an existential quantifier. As entailment checks correspond to universal quantification, this intuitively means that for permissibility we need to test the validity of a $\exists\forall$ formula, instead of a $\forall$ formula for entailment. So we add a quantifier alternation step, which typically does come at the price of increased complexity. This line of thought also immediately provides an intuitive answer to question (2), namely "yes but only if the underlying logic contains $\exists\forall$ quantification".

Of course, both these answers are only approximate and only speak in broad terms. Whether each is to be answered with "yes" or "no" depends on the precise form of the logic, and on what kind of blow-up we are willing to tolerate. To make matters concrete, we now consider three particular logics, namely first-order predicate logic, description logic (more specifically a particular version of $\mathcal{ALC}$) and propositional logic (i.e., first-order predicate logic given a finite universe and without quantification). We start with the latter.

In what follows, say we need to check whether formula f is permitted in DB under circumstances $\mathsf{CaseDesc}$. We abstract from the complications entailed by maintaining hierarchical consistency, and assume that for crt, it holds that $\mathsf{must\text{-}agree}(\mathsf{crt}) = \emptyset$.

Theorem 6. *For propositional logic, deciding permissibility is Σ_2^p-complete.*

Proof sketch. The set $\Sigma_2^p = \mathbf{NP}^{\mathbf{NP}}$, so containment is shown by guessing a supporting set and verifying its properties using an **NP** oracle. For the hardness we encode an QBF formula $\exists x \forall y : \phi(x, y)$ in permissibility request for case law database. We do this by encoding all possible values for x in the database and asking for the permissibility of $\phi(x, y)$. Details can be found in [6].

As entailment testing in propositional logic is only **coNP**-complete, Theorem 6 answers question (1) with "yes", and answers question (2) with "no, unless we are willing to tolerate worst-case exponentially large formulas".

Theorem 7. *Permissibility is equivalent to satisfiability of a formula whose size is polynomial in the size of DB, CaseDesc, and f for*
(1) first-order logic.
*(2) the description logic $\mathcal{ALC}$ with concept constructors **fills** and **one-of** by role constructors **role-and**, **role-not**, **product**, and **inverse**.*[2]

Proof sketch. The result in [12] shows equality of expressivity of first-order logic with at most two free variables. Thus we construct a suitable formula for the first part. We do this by using existantial quantification in order to choose a warranted supporting set and then design the formula such that it is satisfiable if and only if the consistency properties of the case holds that can be constructed from that supporting set (i.e., the case potentially output by Algorithm 1). All parts that are not choosen by the existantial quantifier will be equivalent to $\top$. Details can be found in [6].

Norm Extraction. As seen in Section 3.5, privacy cases induce normative rules. The format of rules gives the advantage that these are easy to enforce and bridge the gap towards privacy policies. As shown by Theorem 5 we extract a norm for every case in the database. The algorithm is postponed to the long version [6]. It basically turns the proof of Theorem 5 into an algorithm transforming the logical formula of the case's facts.

Let f be the size of the biggest formula in the leaves of C and n the number of nodes in C. Then the size of the norm can become $\mathcal{O}(2^f \cdot n + |\mathsf{pre}_C|)$. The computation needs operations linear in that size.

5 Logic Selection

For modeling purposes as well as for computational purposes the choice of logic is, of course, of paramount importance. The only hard requirement ("must have") that the logic, $\mathcal{L}$, must meet is:

(i) **Sufficient expressivity** to tackle our framework and reasoning tasks. Precisely, the minimal requirement is for $\mathcal{L}$ to provide a language $\mathcal{L}_{\mathcal{F}}$ for formulas, with reasoning support for tests of the form (a) $\bigwedge_{\phi \in \Phi} \models \bot$ and (b)

[2] For details on this instance of $\mathcal{ALC}$, please consult [12].

$\bigwedge_{\phi \in \Phi} \models \psi$: These are the only tests our reasoning tasks demand from the underlying logic. If $\mathcal{L}_{\mathcal{F}}$ is closed under conjunction and contains $\bot$ (as will be the case in our logic of choice), the requirement simply becomes to be able to test whether $\phi \models \psi$.

The soft requirements ("nice to have") on the logic are:

(ii) **Suitable for modeling real-world phenomena and knowledge**, ideally an established paradigm for such modeling tasks.
(iii) **Decidability, and as low complexity as possible**, of the relevant reasoning (e.g., satisfiability checks; cf. (i)).
(iv) **Effective tool support** established and available.

What we have just outlined is essentially a "wanted poster" for *description logic (DL)* [4]. This is a very well investigated family of fragments of first-order logic, whose mission statement is to provide a language for modeling real-world phenomena and knowledge (ii), while retaining decidability and exploring the trade-off of expressivity vs. complexity (iii). Effective tool support (iv) has been an active area for two decades. Every DL provides a language to describe "axioms", and even the most restricted DLs make it possible to answer queries about the truth of an axiom relative to a conjunction of axioms, which is exactly the test we require.

We briefly consider the description logic *attributive concept language with complements*, for short $\mathcal{ALC}$ [27,5], which is widely regarded as the canonical "basic" description logic variant (most other DLs extend $\mathcal{ALC}$, in a variety of directions). Description logic is a form of predicate logic that considers only 1-ary and 2-ary predicates, referred to as *concepts* and *roles*, respectively. Assuming a set N_C of concept names and a set N_R of role names, DL makes it possible to construct *complex concepts*, which correspond to a particular subset of predicate-logic formulas with exactly one free variable. For $\mathcal{ALC}$, the set of complex concepts is the smallest set s.t.

1. $\top, \bot$ and every concept name $A \in N_C$ are complex concepts, and
2. if C and D are complex concepts and $r \in N_R$, then $C \sqcap D$, $C \sqcup D$, $\neg C$, $\forall r.C$, and $\exists r.C$ are complex concepts.

Here, $\sqcap$ denotes concept intersection (logical conjunction), $\sqcup$ denotes concept union (logical disjunction), and $\neg C$ denotes concept complement (logical negation). $\forall r.C$ collects the set of all objects x such that, whenever x stands in relation r to y, $y \in C$. Similarly, $\exists r.C$ collects the set of all objects x such that there exists y where x stands in relation r to y and $y \in C$.

$\mathcal{ALC}$ allows *concept inclusion* axioms, of the form $C \sqsubseteq D$, where C, D are complex concepts, meaning that C is a subset of D (universally quantified logical implication). $\mathcal{ALC}$ furthermore allows *assertional* axioms, of the form $x : C$ or $(x, y) : r$, where C is a complex concept, r is a role, and x and y are individual names (i.e., constants). An $\mathcal{ALC}$ knowledge base consists of finite sets of concept inclusion axioms and assertional axioms (called the *TBox* and *ABox* respectively), interpreted as conjunctions. The basic reasoning services provided by $\mathcal{ALC}$ (and most other DLs) are testing whether a knowledge base KB is satisfiable, and testing whether $\mathsf{KB} \models \phi$ where ϕ is an axiom. These decision problems are decidable, and more precisely, ExpTime-complete for $\mathcal{ALC}$.

For our purposes, we can assume as our formulas $\mathcal{L}_{\mathcal{F}}$ conjunctions of axioms, i.e., the smallest set that contains $\bot$, all axioms of the underlying DL (e.g., $\mathcal{ALC}$), as well as $\phi \wedge \psi$ if ϕ and ψ are members of $\mathcal{L}_{\mathcal{F}}$. In order to test whether $\phi \models \psi$, we then simply call the DL reasoning service "$\phi \models \psi_i$?" for every conjunct ψ_i of ψ and return "yes" iff all these calls did. In other words, we may use conjunctions of DL axioms in the knowledge base, case descriptions, and proof tree nodes.

6 Conclusion

In this paper, we introduced PriCL, the first framework for automated reasoning about case law. We showed that it complies with natural requirements of consistency. Moreover, we showed a tight connection between privacy case law and the notion of norms that underlies existing rule-based privacy research. We identified the major reasoning tasks such as checking the case law database for consistency, extracting norms and deducing whether an action is legal or not. For all these tasks, we gave algorithms deciding them and we did an analysis that leads to $\mathcal{ALC}$ as a suitable instantiation for the logic.

Acknowledgements. We want to thank the anonymous reviewer for their valuable feedback. We tried to incorporate the feedback as much as possible. Due to space constraints, parts of the feedback was only used for the long version [6].

This work was supported by the German Ministry for Education and Research (BMBF) through funding for the Center for IT-Security, Privacy and Accountability (CISPA).

References

1. Anderson, A.: A comparison of two privacy policy languages: EPAL and XACML (2005)
2. Annas, G.J.: Hipaa regulations-a new era of medical-record privacy? New England Journal of Medicine 348(15), 1486–1490 (2003)
3. Ashley, P., Hada, S., Karjoth, G., Powers, C., Schunter, M.: Enterprise privacy authorization language (EPAL 1.2). Submission to W3C (2003)
4. 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)
5. Baader, F., Horrocks, I., Sattler, U.: Description Logics. In: Handbook of Knowledge Representation, ch. 3, pp. 135–180. Elsevier (2008)
6. Backes, M., Bendun, F., Hoffman, J., Marnau, N.: PriCL: Creating a Precedent. A Framework for Reasoning about Privacy Case Law (Extended Version) (2015), `http://arxiv.org/abs/1501.03353`
7. Backes, M., Karjoth, G., Bagga, W., Schunter, M.: Efficient comparison of enterprise privacy. In: Proc. of Symposium on Applied Computing, pp. 375–382. ACM (2004)
8. Barth, A., Datta, A., Mitchell, J.C., Nissenbaum, H.: Privacy and contextual integrity: Framework and applications. In: Proc. of S&P, p. 15. IEEE (2006)

9. Barth, A., Mitchell, J.C., Datta, A., Sundaram, S.: Privacy and utility in business processes. In: CSF, vol. 7, pp. 279–294 (2007)
10. Basin, D., Klaedtke, F., Marinovic, S., Zălinescu, E.: Monitoring compliance policies over incomplete and disagreeing logs. In: Qadeer, S., Tasiran, S. (eds.) RV 2012. LNCS, vol. 7687, pp. 151–167. Springer, Heidelberg (2013)
11. Basin, D.A., Klaedtke, F., Müller, S., Pfitzmann, B.: Runtime monitoring of metric first-order temporal properties. In: Proc. of FSTTCS, pp. 49–60 (2008)
12. Borgida, A.: On the relative expressiveness of description logics and predicate logics. Artificial Intelligence 82(1), 353–367 (1996)
13. Breaux, T.D., Antón, A.I.: Analyzing regulatory rules for privacy and security requirements. IEEE Trans. on Software Engineering 34(1), 5–20 (2008)
14. Cavoukian, A.: Privacy by design. Report of the Information & Privacy Commissioner Ontario, Canada (2012)
15. Datta, A., Blocki, J., Christin, N., DeYoung, H., Garg, D., Jia, L., Kaynar, D., Sinha, A.: Understanding and protecting privacy: formal semantics and principled audit mechanisms. In: Jajodia, S., Mazumdar, C. (eds.) ICISS 2011. LNCS, vol. 7093, pp. 1–27. Springer, Heidelberg (2011)
16. DeYoung, H., Garg, D., Kaynar, D., Datta, A.: Logical specification of the glba and hipaa privacy laws. CyLab, p. 72 (2010)
17. Duma, C., Herzog, A., Shahmehri, N.: Privacy in the semantic web: What policy languages have to offer. In: Proc. of POLICY, pp. 109–118. IEEE (2007)
18. European Commission. General data protection regulation, `http://ec.europa.eu/justice/data-protection/document/review2012/com_2012_11_en.pdf`
19. Garg, D., Jia, L., Datta, A.: Policy auditing over incomplete logs: theory, implementation and applications. In: Proc. of CCS, pp. 151–162. ACM (2011)
20. Gürses, S., Gonzalez Troncoso, C., Diaz, C.: Engineering privacy by design. Computers, Privacy & Data Protection (2011)
21. Karat, J., Karat, C.-M., Bertino, E., Li, N., Ni, Q., Brodie, C., Lobo, J., Calo, S., Cranor, L., Kumaraguru, P., Reeder, R.: Policy framework for security and privacy management. IBM Journal of Research and Development 53(2), 4 (2009)
22. Lämmel, R., Pek, E.: Understanding privacy policies. Empirical Software Engineering 18(2), 310–374 (2013)
23. Maffei, M., Pecina, K., Reinert, M.: Security and privacy by declarative design. In: Proc. of CSF, pp. 81–96. IEEE (2013)
24. Ni, Q., Bertino, E., Lobo, J., Brodie, C., Karat, C.-M., Karat, J., Trombeta, A.: Privacy-aware role-based access control. Proc. of TISSEC 13(3), 24 (2010)
25. Office for Civil Rights, U.S. Department of Health and Human Services. Summary of the HIPAA privacy rule (2003)
26. Oh, S.E., Chun, J.Y., Jia, L., Garg, D., Gunter, C.A., Datta, A.: Privacy-preserving audit for broker-based health information exchange. In: Proc. of Data and Application Security and Privacy, pp. 313–320. ACM (2014)
27. Schmidt-Schauß, M., Smolka, G.: Attributive concept descriptions with complements. Artificial Intelligence 48(1), 1–26 (1991)
28. Sen, S., Guha, S., Datta, A., Rajamani, S.K., Tsai, J., Wing, J.M.: Bootstrapping privacy compliance in big data systems. In: Proc. of S& P
29. Tschantz, M.C., Datta, A., Wing, J.M.: Formalizing and enforcing purpose restrictions in privacy policies. In: Proc. of S& P, pp. 176–190. IEEE (2012)
30. United States Congress. Financial services modernization act of 1999 (2010)
31. United States federal law. Children's Online Privacy Protection Act (1998)