Logic of PriCL requires...

truth/falseness (denoted respectively as $\top$ and $\bot$), conjunction (denoted $\land$), entailment (denoted $A \models B$ if formula $A$ entails formula $B$), and monotonicity regarding entailment, i.e., if $A \models B$ then $A \land C \models B$ for any formula $C$. As

**Definition 1 (Case).** A case $C$ is a tuple $(df, CaseDesc, ProofTree, crt)$ s.t.
- $df$ is a formula that we call the decision formula of $C$.
- $CaseDesc$ is a formula describing the case's circumstances.
- $ProofTree$ 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 $l \in \{Axiom, Assess\} \cup \{Ref(i) \mid i \in C_I\}$. Leaf formulas $l$ are additionally associated with a prerequisite formula $pre$. For leaves annotated with $Axiom$, we require that $pre = l$.
- $crt \in Courts$.

“pre $\rightarrow$ fact” is a description of human judgment, not logical implication

“pres_c” and “facts_c” give all prerequisites/facts in a tree given $CaseDesc \models pre$. Type: tree $\rightarrow$ set(p/f)

Subcases are subtrees with around a node $n$ with $df_sub \leftarrow n$

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

df is always necessarily of form $is\_legal\_action(a)$ because of the nature of privacy law

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

**Definition 4 (Case Consistency).** Let $C = (df, CaseDesc, ProofTree, crt)$ 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) $KB_W \land CaseDesc \not\models \bot$

(ii) $KB_W \land CaseDesc \models pres_C$

(iii) $KB_W \land CaseDesc \land facts_C \not\models \bot$

(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
Explain intuition behind OR

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

- \( C \) is a set of cases. We will also write \( C \in DB \) for \( C \in C \).
- \( \mu : C \to C_t \) is an injective function such that \( C \) is closed under \( \mu \). In the following we will also write \( \text{Ref}(D) \) for \( \text{Ref}(i) \) if \( \mu(D) = i \).
- Let \( \prec_{\text{ref}} := \{(C, D) \mid D \text{ contains a } \text{Ref}(C) \text{ node}\} \) and \( \leq_t \) is an order that we call time order of the cases. It has to hold:

\[
\text{must-agree} \subseteq \prec_{\text{ref}} \subseteq \text{may-ref} \subseteq \leq_t \subseteq C \times C
\]

- \( U \) specifies the unwarranted nodes, i.e., \( U : C \to N \) is function such that
  - \( N \) is a subset of the nodes labelled with \( \text{Assess} \) or \( \text{Ref} \) in the cases \( C \).
  - The set increases monotonic, i.e., \( C \leq_t D \implies U(C) \subseteq U(D) \).

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

Explain must-agree vs may-ref / ratio decidendi vs obiter dicta

To be warranted, a case must not require prerequisites

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

A reference is correct if there are shared decision requirements on a node and shared prerequisites

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

Cases conflict if their facts or prerequisites contradict and they must agree (because of their df's)
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) $KB_W \land \text{pres}_{C_1} \land \text{pres}_{C_2} \not\models \bot$

(ii) $KB_W \land \text{facts}_{C_1} \land \text{facts}_{C_2} \models \bot$

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

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

Probably just gloss over this as it closely matches intuition and is way too long:

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

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

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

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

(iv) warrants consistently if for every $C$ holds: $U(C')$ contains all $\text{Ref}(D)$ 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.

Deduce = must follow from existing law; Permit = could follow or opposite could follow

Definition 10 (Deducibility and Permissibility). Let $DB = (C, \leq_t, \text{must-agree}, \text{may-ref}, \mu, U)$ be a consistent CLD, and $f$ a formula. We say that $f$ is permitted in $DB$ under circumstances CaseDesc and court $\text{crt}$ if there exists a case $C = (f, \text{CaseDesc}, \text{ProofTree}, \text{crt})$ 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 $\text{crt}$ if $\neg f$ is not permitted under CaseDesc and $\text{crt}$. 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 $\text{crt}$ if there exists a set of cases $\{C_f = (f, \text{CaseDesc}, \text{ProofTree}_f, \text{crt}) \mid f \in F\}$ such that every 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 $\leq_t$).

This doesn't actually seem that important but does a neat trick in (ii)

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

(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.
A supporting set is the facts/pres from which permissibility arises.

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

1. $KB_W \land CaseDesc \models \bigwedge_{(pre\rightarrow fact) \in A} pre$
2. $KB_W \land CaseDesc \land \bigwedge_{(pre\rightarrow fact) \in A} fact \models f$
3. $KB_W \land CaseDesc \land \bigwedge_{(pre\rightarrow fact) \in A} fact \not\models \bot$

This trivial follows from definitions but “suggests an algorithm” if you love exponential time

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

This really just clarifies the reasons for the specificity of Thm 2… moving on.

**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 $DB \cup \{C_i\}$.

I think this last one is pretty obvious.

**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 $\in must-agree(crt)$.

Norms = implicit laws

**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 $\phi$. The norm is a positive norm, denoted $\phi^+$, if $p = is_legal_action(a)$ and a negative norm, denoted $\phi^-$, if $p = \neg is_legal_action(a)$. A norm $\phi$ decides $p$ given $f$ if $KB_W \land f \models \phi$.

Norms can be found...
Theorem 5. Let DB be a consistent privacy case law database and $C = (df, CaseDesc, ProofTree, crt) \in DB$. Then there is a norm $\phi$ that decides $df$ given CaseDesc. In particular, there are formulas $\phi_W, \phi_S$ such that $is\_legal\_action(a)$ does not occur in these formulas and

1. $\text{facts}_C \Rightarrow \phi_W \land (\phi_S \Rightarrow df)$
2. $\phi_W \land (\phi_S \Rightarrow df) \Rightarrow df$

And cases restructured around them...

Corollary 1 (Normal forms). Let $DB = (C, \leq_t, \text{must-agree}, \text{may-ref}, \mu, U)$ 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 $\phi_w$ 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 $(C\setminus\{C\} \cup \{N(C)\}, \leq_t)$ is also consistent (where $N(C)$ is placed at the position of $C$ w.r.t. $\leq_t$).

This is determined by a brute force “Algorithm 1: Permissibility” - nevertheless an algorithm though

sum$^p_2$ means “NP with NP oracle”

Theorem 6. For propositional logic, deciding permissibility is $\Sigma^p_2$-complete.

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 $ALC$ with concept constructors $fills$ and $one\_of$ by role constructors $role\_and$, $role\_not$, $product$, and $inverse$.

The logic to use is “attributive concept language with complements” because of nice properties...