PriCL notes by Calvin

A Brief Introduction to Law: Common Law v Civil Law, Precedence

Logic of PriCL requires…

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

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 $\mathsf{pre} = l$.*
- *$\mathsf{crt} \in \mathsf{Courts}$.*

"pre → fact" is a description of human judgment, not logical implication

"pres_c" and "facts_c" give all prerequisites/facts in a tree given CaseDesc |= pre. Type: tree → set(p/f)

Subcases are subtrees with around a node n with df_sub ← n

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.*

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 $\mathsf{Actions}$, 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 $\mathsf{CaseDesc}$.*

Definition 4 (Case Consistency). *Let $C = (\mathsf{df}, \mathsf{CaseDesc}, \mathsf{ProofTree}, \mathsf{crt})$ 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) $\mathsf{KB}_W \wedge \mathsf{CaseDesc} \not\models \bot$ *(ii)* $\mathsf{KB}_W \wedge \mathsf{CaseDesc} \models \mathsf{pres}_C$

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

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

Explain intuition behind OR

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

- $\mathbf{C}$ *is a set of cases. We will also write* $C \in \mathit{DB}$ *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* $\mathit{Ref}(D)$ *for* $\mathit{Ref}(i)$ *if* $\mu(D) = i$.
- *Let* $<_{\mathit{ref}} := \{(C, D) \mid D$ *contains a* $\mathit{Ref}(C)$ *node*$\}$ *and* $\leq_t$ *is an order that we call* time order *of the cases. It has to hold:*

$$\begin{matrix} \mathit{must\text{-}agree} \subseteq \\ <_{\mathit{ref}} \subseteq \end{matrix} \mathit{may\text{-}ref} \subseteq \leq_t \subseteq \mathbf{C} \times \mathbf{C}$$

- 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* $U(\mathit{DB}) := \bigcup_{C \in \mathbf{C}} U(C)$.

tuitively, $\mathsf{may\text{-}ref}(C_1, C_2)$ 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.

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* $(\mathit{df}, \mathit{CaseDesc}, \mathit{ProofTree}, \mathit{crt})$ *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* $\bot$.

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* $C = (\mathit{df}, \mathit{CaseDesc}, \mathit{ProofTree}, \mathit{crt})$ *a case in* DB. *A leaf node* $\mathit{pre} \to \mathit{fact}$ *in* $\mathit{ProofTree}$ *annoted with* $\mathit{Ref}(D)$ references correctly *if* $D_u = (\mathit{fact}, \mathit{CaseDesc}_D, \mathit{ProofTree}_D, \mathit{crt}_D)$ *is a warranted subcase of a case* $D \in \mathit{DB}$ *w.r.t.* $U(C)$, $\mathit{may\text{-}ref}(C, D)$ *holds and* $\mathit{KB}_W \wedge \mathit{pre} \models \mathit{pres}_D$. C references correctly *if all its leaves annoted with* $\mathit{Ref}(D)$ *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) $\mathsf{KB}_W \wedge \mathsf{pres}_{C_1} \wedge \mathsf{pres}_{C_2} \not\models \bot$ *(ii)* $\mathsf{KB}_W \wedge \mathsf{facts}_{C_1} \wedge \mathsf{facts}_{C_2} \models \bot$

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

A case C is in conflict with DB if there is a $D \in \mathsf{DB}$ 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 $\mathsf{DB} = (\mathbf{C}, \leq_t, \mathsf{must\text{-}agree}, \mathsf{may\text{-}ref}, \mu, U)$ is*

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

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

(iii) hierarchically consistent *if every $C \in \mathsf{DB}$ 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.*

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

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

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

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

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, $\mathsf{CaseDesc}$ a case description and crt a court. The following holds:*

1. *$C \in \mathsf{DB}$ with warranted node $f \Rightarrow \exists \mathcal{A}$ that supports f*
2. *f is permitted (under circumstance $\mathsf{CaseDesc}$ 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*

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

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 $\mathsf{CaseDesc}$ and court crt. The same holds for $\{f, \neg f\}$ if $\mathsf{crt} \in \mathsf{must\text{-}agree}(\mathsf{crt})$.*

Norms = implicit laws

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

Norms can be found...

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

And cases restructured around them…

Corollary 1 (Normal forms). *Let $DB = (\mathbf{C}, \leq_t, \mathsf{must\text{-}agree}, \mathsf{may\text{-}ref}, \mu, U)$ be a privacy case law database, $C = (\mathsf{df}, \mathsf{CaseDesc}, \mathsf{ProofTree}, \mathsf{crt}) \in DB$ 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 $\phi_S \Rightarrow \mathsf{df}$ 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. $\leq_t$).*

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 Σ_2^p-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 $\mathcal{ALC}$ with concept constructors **fills** and **one-of** by role constructors **role-and**, **role-not**, **product**, and **inverse**.*[2]

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