CHiRiSM and Probabilistic Argumentation Logic

Jon Sneyers\textsuperscript{1}, Danny De Schreye\textsuperscript{1}, and Thom Frühwirth\textsuperscript{2}

\textsuperscript{1} Dept. of Computer Science, KU Leuven, Belgium\linebreak jon.sneyers@cs.kuleuven.be, danny.deschreye@cs.kuleuven.be\linebreak \textsuperscript{2} University of Ulm, Germany\linebreak thom.fruehwirth@uni-ulm.de

Abstract. Riveret et al. proposed a framework for probabilistic legal reasoning. Their goal is to determine the chance of winning a court case, given the chances of the judge accepting certain claims and legal rules. We tackle the same problem by defining and implementing a new formalism, called probabilistic argumentation logic (PAL). We implement PAL in CHiRiSM, and discuss how it can be seen as a probabilistic generalization of Nute’s defeasible logic. Not only does this provide an automation of the — only hand-performed — computations in Riveret et al, it also provides a solution to one of their open problems: a method to determine the initial probabilities from a given body of precedents.

1 Introduction

Riveret et al.\cite{riveret2010} proposed a framework of probabilistic legal reasoning based on the argumentation framework of Prakken et al.\cite{prakken2001}, which provides a dialectical proof theory in the formal setting of Dung\cite{dung1995a}. Their goal is to determine the chance of winning a court case, given the probabilities of the judge accepting certain claimed facts to be valid and legal rules to be applicable.

Roth et al.\cite{roth2011} tackle a similar problem, but with the focus on finding legal strategies for the involved parties to maximize their chances of winning. They propose a rather complex framework, based on a logic layer, an argument layer, a dialectical layer, a procedural layer, and finally a probabilistic weighting.

To our knowledge, none of these approaches have been implemented so far. Both papers only contain a hand-performed computation to illustrate their approach on an example. Such an implementation of an ‘argument assistance system’ is left explicitly as future work in\cite{riveret2010}.

Another issue is how to determine or verify the probabilities, which are assumed to be known in advance. Riveret et al.\cite{riveret2010} suggest that maybe somehow a statistical analysis of how the judge has decided in the past could be performed, but they again leave this issue to future work.

In this paper we tackle the problem by defining a new formalism, called probabilistic argumentation logic, which can be seen as a probabilistic generalization of defeasible logic with explicit underlying assumptions.

We formalize our approach and implement it in CHiRiSM\cite{chirms}. CHiRiSM is a rule-based probabilistic logic programming language based on CHR\cite{chr} in the
host language PRISM [7]. For reasons of space we will omit an introduction to these formalisms; we assume the reader is already familiar with them. We refer the reader to [6, 8, 9] for CHR and to [5] for CHRIISM. In a companion paper [10], an introduction to CHRIISM can be found, as well as a discussion of the advantages and disadvantages of CHRIISM in this setting.

Our implementation provides an automation of the probability computations that were hand-performed in [1]. The built-in learning algorithm of CHRIISM also provides a solution to the open problem of determining the probabilities.

The remainder of this paper is organized as follows. In Section 2 we introduce the approach of [1] and the running example in their paper. We show how this example can be encoded in CHRIISM. Section 3 introduces our new formalism, probabilistic argumentation logic. We show that it generalizes Nute’s defeasible logic [11] and we give an implementation in CHRIISM. In Section 4 we briefly discuss learning. Finally we conclude in Section 5.

2 The mad cow example and dialogue games

The running example in [1] is the following. John, the proponent, wants to sue Henry, the opponent, claiming compensation for the damage that Henry’s cow caused to him when he drove off the road to avoid the cow. John argues that an animal’s owner has to pay damages caused by their animal, that Henry is the owner of the cow, and that the accident was caused by the need to avoid the cow (argument A). Henry can counterattack in various ways: he can claim that the damage was due to John’s negligence (he did not pay sufficient attention to crossing animals) – argument B – or that it was a case of force majeure: the cow suddenly went crazy and crossed into the street – argument C. The last objection could be replied to by using the debated rule that only exogenous events count as force majeure, and the cow’s madness is endogenous (argument D).

[1] assumes an abstract argumentation framework [3], which consists of a set of arguments and a binary “defeats” relation. They then define the notion of a dialogue game which captures the rules of legal argumentation. A dialogue is a sequence of abstract arguments, in which the proponent and the opponent alternate, each argument defeats the previous argument, and the proponent cannot repeat arguments and his arguments have to strictly defeat the previous argument of the opponent.

Moreover, each of the arguments has some given “construction chance”, which is the probability that the judge will actually accept the argument. The aim is to estimate the overall chance that the case is won.

2.1 Encoding of dialogue games in CHRIISM

In general, [1] assume an abstract argumentation framework [3], which consists of a set of arguments and a binary “defeats” relation. Argument X strictly defeats argument Y if X defeats Y and Y does not defeat X. For example, argument A and B defeat one another, while argument D strictly defeats argument C.
Moreover, each of the arguments has some given “construction chance”, which is the probability that the judge will actually accept the argument. The aim is to estimate the overall chance that the case is won.

They consider dialogue games which capture the rules of legal argumentation. A dialogue is a sequence of arguments, in which the proponent and the opponent alternate, with the following conditions:

- The proponent cannot repeat arguments;
- Every opponent argument defeats the previous (proponent) argument;
- Every proponent argument strictly defeats the previous argument;
- If a player cannot make a move, he loses.

The first argument is the main claim of the proponent. If the proponent wins all possible dialogues, he wins the case. Otherwise the opponent wins. Obviously, the outcome depends on which arguments are actually accepted by the judge.

In the mad cow example there are two possible dialogues if the judge accepts all arguments: [A,B] (in which the proponent loses) and [A,C,D] (in which the proponent wins). So if the judge accepts all arguments, the proponent loses (since he has no winning strategy in case the opponent uses argument B).

We can directly encode the rules of the dialogue game in CHRiSM, as follows. We use a dummy constraint predicate begin/0 to initialize a dialogue:

\[
\text{begin} \iff \text{init_defeats}, \text{dialogue}([{}]).
\]

The auxiliary predicate init_defeats/0 initializes the “defeats” relation:

\[
\text{init_defeats} \iff \text{defeats(arg\ A,arg\ B)}, \text{defeats(arg\ B,arg\ A)}, \\
\text{defeats(arg\ A,arg\ C)}, \text{defeats(arg\ C,arg\ A)}, \\
\text{defeats(arg\ D,arg\ C)}.
\]

The “strictly defeats” relation is derived from the “defeats” relation:

\[
\text{defeats(A,B)} \implies \text{strictly_defeats(A,B)}.
\]

\[
\text{defeats(B,A)} \setminus \text{strictly_defeats(A,B)} \iff \text{true}.
\]

The main constraint predicate is dialogue/1, which contains a (reversed) list representing a (partial) dialogue. For example, a dialogue [A,C,D] would be encoded as dialogue([p-argD,o-argC,p-argA]). Note the reversed order (for easy access to the last element) and the “p-” and “o-” tags to denote the player.

First of all, we make sure that the construction chances are taken into account. If the construction chance of some argument is \( p \), then we simply prune away a partial dialogue that ends with that argument with probability \( 1 - p \). For example, suppose the construction chance of argument A is 0.9, then we remove a dialogue ending with A with probability 0.1. These are the values used in [1]:

\[
0.1 \ \text{?? dialogue([_argA|_])} \iff \text{true}.
\]

\[
0.6 \ \text{?? dialogue([_argB|_])} \iff \text{true}.
\]

\[
0.9 \ \text{?? dialogue([_argC|_])} \iff \text{true}.
\]

\[
0.85 \ \text{?? dialogue([_argD|_])} \iff \text{true}.
\]
Now we add a rule for the proponent to make his main claim, argument A:

\[ \text{dialogue}([]) \Rightarrow \text{dialogue}([p\text{-arg}A]). \]

The rest of the dialogue is constructed according to the above conditions:

\[ \begin{align*}
\text{dialogue}([p\text{-A}|D]), & \text{defeats}(B,A) \Rightarrow \text{dialogue}([o\text{-B},p\text{-A}|D]). \\
\text{dialogue}([o\text{-A}|D]), & \text{strictly_defeats}(B,A) \Rightarrow \text{dialogue}([p\text{-B},D] \mid \text{dialogue}([p\text{-B},o\text{-A}|D])).
\end{align*} \]

Now comes a tricky part: if a partial dialogue has been extended (but only after it has been extended in all possible ways), we can discard it. If it could not be extended, we have a final dialogue, which was won by the last player. One way to implement this is as follows: (exploiting the refined operational semantics and the passive pragma)

\% dialogue was extended: remove prefix
\[ \text{dialogue}([_\mid D]) \# \text{passive} \setminus \text{dialogue}(D) \Leftrightarrow \text{true}. \]

\% dialogue was not extended: last player wins
\[ \text{dialogue}([X\text{-}_\mid \_]) \Rightarrow \text{winner}(X). \]

Finally, as soon as there is one dialogue which is won by the opponent, then the proponent cannot be a winner.

\[ \text{winner}(o) \setminus \text{winner}(p) \Leftrightarrow \text{true}. \]

In [1], several pages are used to calculate the probability that the proponent wins the case. In CHRIStM, we can simply use the following query:

\[ ?- \text{prob begin} \Rightarrow \text{winner}(p). \]

Probability of begin \Rightarrow winner(p) is: 0.494100000000000

Where did the dialog go? In the encoding of the arguments in Section 2, we abstract away the turn-based dialog. We impose no total order on the arguments, nor strict alternation between the players; in fact, we do not even record which party (proponent or opponent) makes which claim. The essential structure of the reasoning is left intact though.

The rules were written as if both the proponent and the opponent make all of their claims and arguments simultaneously at the start. In order to model more accurately that some arguments will only be put forward as a “reaction” to the acceptance of other statements, we can also write the rules in a different way, for example, instead of always claiming that the cow was mad:

\[ 0.2 \Rightarrow \text{begin} \Rightarrow \text{accept}(e,[\_]). \]

the opponent Henry only makes that claim if the judge is tempted to make him compensate John’s damages:
0.2 ?? accept(c,_) == accept(e,[]).

Similarly, instead of always insisting that cow madness is endogenous:

0.3 ?? begin == accept(f,[]).

the proponent John only makes that claim if the judge has actually accepted that the cow was mad:

0.3 ?? accept(e,_) == accept(f,[]).

With respect to the outcome of the case (whether or not statement c gets accepted), there is no difference between both ways of encoding. The only difference is that the “put everything on the table at once” approach can introduce irrelevant claims, while the “reactive” approach more closely models the dialog, since claims are only triggered when they are needed.

2.2 CHRiSM encoding of the mad cow example

In the above, we described a CHRiSM encoding of dialogue games with abstract arguments, in which the arguments are treated as black boxes that only interact through the “defeats” relation. We now proceed with a finer level of granularity.

The arguments in [1] consist of premises and rules, which each have a probability of being accepted by the judge. For example, argument A consists of the premises that Henry owns the cow (a), and that the accident was caused by the need to avoid the cow (b), together with the rule “a \& b \rightarrow c”, where c stands for “Henry has to compensate damages”. If a will certainly be accepted, b is accepted with a probability of 0.9, and the rule “a \& b \rightarrow c” is certainly accepted, then the overall construction chance of argument A is 1 \times 0.9 \times 1 = 0.9.

We will call both premises and conclusions “statements” and use ground Prolog terms to denote them. The auxiliary predicate neg/2 simply negates a literal in a way that avoids double negations.

We use a dummy predicate begin/0 (to be used as the initial goal). The main constraint predicate is accept/2, which indicates that the judge conditionally accepts some statement: accept(S,C) denotes that statement S is accepted if all conditions C (a Prolog list of statements) hold. If C is the empty list, the statement is unconditionally accepted; otherwise the acceptance of the statement can still be retracted if one of the conditions turn out to be unacceptable. If a statement is already accepted with conditions A, then it is redundant to also accept it with stronger conditions B \supseteq A.

accept(X,A) \ accept(X,B) <=> subset(A,B) | true.

The above rule implies a set semantics for unconditionally accepted statements.

A condition is redundant if it is implied by the other conditions:

accept(A,B) \ accept(X,C) <=> select(A,C,D), subset(B,D) | accept(X,D).
If a statement $Y$ was accepted with conditions $B$, but one of those conditions is contradicted (“undercut”) by a statement $X$ with weaker conditions $A$ that are implied by $B$, then the acceptance of $Y$ has to be retracted — we use a simpagation rule to remove the $\text{accept}(Y,B)$ constraint:

$$\text{accept}(X,A) \setminus \text{accept}(Y,B) \iff \text{subset}(A,B), \text{neg}(X,NX), \text{member}(NX,B) \mid \text{true}.$$ 

Finally, we allow no contradictions:

$$\text{accept}(X,A), \text{accept}(Y,B) \implies \text{subset}(A,B), \text{neg}(X,Y) \mid \text{fail}.$$ 

Now we encode the premises and the rules of the arguments:

% Argument A (rule r1, premises a and b):
% "If Henry is the owner of the cow (a) and the accident was caused by the need to avoid the cow (b), then Henry has to compensate damages (c)."

1.0 ?? $\text{accept}(a,[])$, $\text{accept}(b,[]) \implies \text{accept}(c, [\text{app(r1)}]).$

0.9 ?? $\text{begin} \implies \text{accept}(a,[]).$

0.9 ?? $\text{begin} \implies \text{accept}(b,[]).$

The rule being used is a defeasible rule, so its conclusion $c$ will be accepted with the condition that the rule is actually applicable ($\text{app(r1)}$). This condition can be “undercut” by arguments B or C.

% Argument B (rule r2, premise d):
% "If John was negligent (d), then r1 is not applicable."

0.8 ?? $\text{accept}(d,[]) \implies \text{accept}(\text{not(app(r1)),}[],[]).$

0.5 ?? $\text{begin} \implies \text{accept}(d,[]).$

For example, the judge could accept $a$, $b$ and $d$ and both rules, to reach the state $\text{accept(c,[app(r1)])}, \text{accept(not(app(r1)),}[],[]).$ Now the undercutting rule removes the conditional acceptance of $c$, because its condition was contradicted.

% Argument C (rule r3, premise e):
% "If the cow was mad (e), it was a case of force majeure. % so r1 is not applicable."

0.5 ?? $\text{accept}(e,[]) \implies \text{accept}(\text{not(app(r1)), [app(r3)])}.\$

0.2 ?? $\text{begin} \implies \text{accept}(e,[]).$

Again, the above rule (r3) is defeasible, so its conclusion can only be accepted with the condition $\text{app(r3)}$, which can be undercut by the final argument:

% Argument D (rule r4, premise f):
% "If the cow’s madness is endogenous (f), then the ‘force majeure’ rule r3 is not applicable."

0.5 ?? $\text{accept}(f,[]) \implies \text{accept}(\text{not(app(r3)),}[],[]).$

0.3 ?? $\text{begin} \implies \text{accept}(f,[]).$

The only thing now left to do, is to resolve the conditions by making assumptions. For example, one possible result of the query $\text{begin}$ is the following:
accept(a, []), accept(b, []), accept(e, []),
accept(c, [app(r1)]), accept(not(app(r1)), [app(r3)])

In this case, both c and not(app(r1)) are conditionally accepted. Although not(app(r1)) undercuts the conditions of c, the “undercut” rule is not applicable (yet) because the condition [app(r3)] is not weaker than the condition [app(r1)]. However, since there is no counter-evidence for app(r3), we can assume this condition to hold, “promoting” not(app(r1)) to an unconditionally accepted statement, which then causes the acceptance of c to be retracted.

To implement this idea, we add the following rules at the end of the program:

begin <=> assume.
assume, accept(X,C) ==> select(A,C,D), accept(A,[]) ; true.
assume <=> true.

The middle rule nondeterministically selects conditions and accepts them. This can either lead to a contradiction (e.g. if there is counter-evidence), which causes backtracking, or succeed.

This concludes our program. We can now compute the desired probability — which took several pages of manual calculations in [1] — with a simple query:

?- prob begin ===> accept(c,[]).
Probability of begin ===> accept(c,[]) is: 0.494100000000000

3 Generalization and Formalization

In order to generalize the running example, we will propose a transformation from an arbitrary probabilistic legal argumentation logic $A$ — a notion that will be introduced in this section — to a CHRiSM program $P_{CHRiSM}(A)$.

3.1 Probabilistic Argumentation Logic

We use $lit(A)$ to denote the set of literals over a set of atomic formulas $A$, i.e. $lit(A) = A \cup \{ \neg a \mid a \in A \}$. We will sometimes denote conjunctions over literals as sets since the order of the conjuncts is irrelevant.

**Definition 1 (Probabilistic argumentation logic).** A probabilistic argumentation logic or PAL is a tuple $(S,A,R,P)$, where $S$ is a set of statements and $A$ is a set of assumptions (with $S \cap A = \emptyset$), $R$ is a set of rules, and $P$ is a function assigning probabilities to each rule, $P : R \mapsto [0,1]$. The rules in $R$ have the following form:

$s_1 \land \ldots \land s_n \Rightarrow c_1 \land \ldots \land c_m$ assuming $a_1 \land \ldots \land a_k$

where the left hand side (the antecedent) is a conjunction of literals ($s_i \in lit(S \cup A)$) which can be empty ($n \geq 0$), the right hand side (the consequent) is a non-empty ($m \geq 1$) conjunction of literals ($c_i \in lit(S \cup A)$), and the part after
“assuming” (the assumption) is a possibly empty ($k \geq 0$) conjunction of assumption literals ($a_i \in \text{lit}(A)$). If the left hand side is empty, the rule is also called a fact and the arrow can be omitted; if the part after “assuming” is empty, the rule is called unconditional and the keyword “assuming” can be omitted.

To illustrate the definition, we now write out the running mad cow example as a formal probabilistic argumentation logic

$$A_{mc} := (\{a,b,c,d,e,f\}, \{\text{app}(r1), \text{app}(r3)\}, R_{mc}, P_{mc})$$

where the rules $R_{mc} = \{r_1, r_2, r_3, r_4, s_a, s_b, s_d, s_e, s_f\}$ are the following:

- $r_1 := a \land b \Rightarrow c$ assuming $\text{app}(r1)$
- $r_2 := d \Rightarrow \neg\text{app}(r1)$
- $r_3 := e \Rightarrow \neg\text{app}(r3)$ assuming $\text{app}(r3)$

$$r_4 := f \Rightarrow \neg\text{app}(r3)$$

$$\forall x \in \{a,b,d,e,f\} : s_x := x$$

and the probabilities $P_{mc}$ are the following:

$$P_{mc} := \{(r_1, 1), (r_2, 0.8), (r_3, 0.5), (r_4, 0.5), (s_a, 1), (s_b, 0.9), (s_d, 0.5), (s_e, 0.2), (s_f, 0.3)\}$$

Now we define an interpretation of a PAL as a set of conditional statements.

**Definition 2 (Conditional statement).** Given a PAL $A = (S, A, R, P)$, a conditional statement is a pair $(s, C)$ with $s \in \text{lit}(S \cup A)$ and $C \subseteq \text{lit}(A)$, such that $C$ is not self-contradictory, that is, there is no $c \in C$ and $\neg c \in C$.

**Definition 3 (Interpretation).** Given a PAL $A$, an interpretation is a set $I$ of conditional statements of $A$ such that if $(s, C_1) \in I$ and $(\neg s, C_2) \in I$, then $C_1 \nsubseteq C_2$ and $C_2 \nsubseteq C_1$.

In other words, the conditional statements are not directly contradictory — although both a statement and its negation can be conditionally accepted at the same time, as long as the assumptions are different.

**Definition 4 (Partial Ordering of Interpretations).** We say an interpretation $I_1$ is smaller than an interpretation $I_2$, denoted $I_1 \leq_i I_2$, if for all conditional statements $(s, c_1) \in I_1$, there is a corresponding conditional statement $(s, c_2) \in I_2$ such that $c_1 \subseteq c_2$.

If $I_1 \leq_i I_2$, then the set of statements in $I_1$ is a subset of the statements in $I_2$, so $I_1$ makes less claims than $I_2$ (which is why we call it smaller), but the claims in $I_1$ are in a sense stronger since they have a weaker condition.

We now define the semantics of a PAL, somewhat inspired by the definitions in [11], but extending them to take the explicit conditions into account, as well as the rule selection (which will be needed to introduce the rule probabilities).
Definition 5 (Compliant Interpretation w.r.t. Rule Selection). Given a PAL $\mathcal{A} = (S, A, R, P)$ and a set of selected rules $R_s \subseteq R$, a compliant interpretation $I$ w.r.t. the selected rules $R_s$ is a minimal (w.r.t. $\leq_i$) interpretation that satisfies the following additional criterion:

- if $R_s$ contains a rule $r = (S_e \Rightarrow C_r$ assuming $A_r)$,
- and $\forall s \in S_r : \exists c : (s,c) \in I$ (the antecedent is conditionally accepted),
- and $\forall a \in A_r : \neg \exists c : (\neg a,c) \in I$ (the assumption is not questioned),
- then for every set $\{(s_1, c_1),\ldots,(s_n, c_n)\} \subseteq I$ such that $S_r = s_1 \land \ldots \land s_n$, it must hold that $\forall x \in C_r : \exists y \subseteq A_r \cup \{c_1,\ldots,c_n\} : (x,y) \in I$ (the consequent is conditionally accepted).

Definition 6 (Valid Interpretation w.r.t. Rule Selection). Given a PAL $\mathcal{A} = (S, A, R, P)$ and a set of selected rules $R_s \subseteq R$, a valid interpretation $I$ w.r.t. the selected rules $R_s$ is a compliant interpretation without gratuitous statements, that is, there is no subset $K \subseteq I$ with $K \neq \emptyset$ such that:

- if $R_s$ contains a rule $r = (S_e \Rightarrow C_r$ assuming $A_r)$,
- and $\forall s \in S_r : \exists c : (s,c) \in I \setminus K$, and $\forall a \in A_r : \neg \exists c : (\neg a,c) \in I$,
- then for every set $\{(s_1, c_1),\ldots,(s_n, c_n)\} \subseteq I \setminus K$ such that $S_r = s_1 \land \ldots \land s_n$, it must hold that $\forall x \in C_r : \forall y \subseteq A_r \cup \{c_1,\ldots,c_n\} : (x,y) \notin K$.

Returning to the mad cow example, consider the rule selection $R_{\text{inc}}$ of all rules. The following is the only valid interpretation w.r.t. $R_{\text{inc}}$:

$$\{(a,\emptyset),(b,\emptyset),(d,\emptyset),(e,\emptyset),(f,\emptyset),(-\text{app(r1)},\emptyset),(-\text{app(r3)},\emptyset)\}$$

This is the only valid interpretation w.r.t. $R' = \{r_1, r_3, r_4, s_b, s_c, s_f\}$:

$$\{(a,\emptyset),(b,\emptyset),(c,\emptyset),(e,\emptyset),(f,\emptyset),(-\text{app(r3)},\emptyset)\}$$

An interpretation like the above, but with $(c, \{\text{app(r1)}\})$ instead of $(c,\emptyset)$ also satisfies the criterion of Def. 5, but it is not compliant because it is not minimal w.r.t. $\leq_i$. The condition for the acceptance of the consequent of applicable rules is allowed to be weaker than (i.e. a subset of) the union of all the conditions arising from the antecedent and assumption. This relaxation serves two goals. Firstly, it means that if a consequent can be derived in different ways such that it would be accepted multiple times with varying conditions, it suffices to have only the weakest conditions in the interpretation. Secondly, because the interpretation has to be minimal w.r.t. $\leq_i$, conditions will only be present in the interpretation to avoid contradiction. The following example illustrates this point.

$$r_1 := \text{bird} \Rightarrow \text{flies} \text{ assuming } \text{normal-bird}$$
$$r_2 := \text{tux} \Rightarrow \text{bird} \land \text{tuxedo-plumage} \text{ assuming } \text{eyes-OK}$$
$$r_3 := \text{tuxedo-plumage} \Rightarrow \text{penguin} \text{ assuming } \text{feathers-make-bird}$$
$$r_4 := \text{penguin} \Rightarrow \neg \text{flies}$$
$$r_5 := \text{tux}$$
Consider the selection of all rules. The following interpretation is compliant:

\\{(tux, \emptyset), (\neg \text{eyes-0K}, \emptyset)\}

but it is not a valid interpretation since the statement \(\neg \text{eyes-0K}\) is trivially gratuitous: there is not even a rule that could derive it.

The following interpretation satisfies the criterion of Def. 5:

\[
\begin{align*}
(tux, \emptyset), & \quad (\text{bird}, \{\text{eyes-0K}\}), \quad (\text{flies}, \{\text{eyes-0K, normal-bird}\}), \\
&(tuxedo-plumage, \{\text{eyes-0K}\}), \quad (\text{penguin}, \{\text{eyes-0K, feathers-make-bird}\}), \\
&\quad (\neg \text{flies}, \{\text{eyes-0K, feathers-make-bird}\})
\end{align*}
\]

but it is not minimal w.r.t. \(\leq_i\); we can relax some conditions to get a minimal interpretation, for example (in this case there are three minimal interpretations)

\[
\begin{align*}
(tux, \emptyset), & \quad (\text{bird}, \emptyset), \quad (\text{flies}, \{\text{normal-bird}\}), \\
&(tuxedo-plumage, \emptyset), \quad (\text{penguin}, \{\text{eyes-0K}\}), \quad (\neg \text{flies}, \{\text{eyes-0K}\})
\end{align*}
\]

Note that some conditions have to be kept in order to avoid a direct contradiction between \text{flies} and \(\neg \text{flies}\). Also note that in the above program, one could replace \(r_4\) with \(\text{penguin} \Rightarrow \neg \text{normal-bird}, \neg \text{flies}\) to get rid of these conditions and have a unique minimal interpretation which contains \((\neg \text{flies}, \emptyset)\).

**Definition 7 (Plausibility of a statement).** Given a PAL \(\mathcal{A} = (S, A, R, P)\), a statement literal \(s \in \text{lit}(S)\) is called plausible w.r.t. a rule selection \(R_s\) if all valid interpretations w.r.t. \(R_s\) contain a conditional statement \((s, c)\) for some \(c\).

**Definition 8 (Acceptability of a statement).** Given a PAL \(\mathcal{A} = (S, A, R, P)\), a statement literal \(s \in \text{lit}(S)\) is called acceptable w.r.t. a rule selection \(R_s\) if it is plausible and there exists a valid interpretation w.r.t. \(R_s\) which contains the conditional statement \((s, \emptyset)\).

In the above example, \text{tux}, \text{bird}, and \text{tuxedo-plumage} are acceptable statements, while \text{flies}, \(\neg \text{flies}\), and \text{penguin} are plausible but not acceptable. All other literals are not even plausible.

The probability \(\text{prob}(R_s)\) of a rule selection \(R_s \subseteq R\) is defined as follows:

\[
\text{prob}(R_s) = \left( \prod_{r \in R_s} P(r) \right) \left( \prod_{r \in R \setminus R_s} 1 - P(r) \right)
\]

which ensures that \(\sum_{R_s \in \mathcal{P}(R)} \text{prob}(R_s) = 1\).

**Definition 9 (Probability of a statement).** Given a PAL \(\mathcal{A} = (S, A, R, P)\), the probability of a statement \(s \in \text{lit}(S)\) is defined as the sum of the probabilities of all rule selections \(R_s \in \mathcal{P}(R)\) in which \(s\) is acceptable.
3.2 Transformation to CHriSM

We now introduce a transformation from an arbitrary probabilistic legal argumentation logic $\mathcal{A} = (S, A, R, P)$ to a CHriSM program $P_{\text{CHriSM}}(\mathcal{A})$. The transformation is a generalization of the example discussed in Section 2.2. We assume the list predicates `member/2`, `subset/2`, and `append/2` (whose first argument is a list of lists) are already defined in the host language (Prolog).

The transformed program starts with the same three rules as in Section 2.2. These rules insure that `accept/2` encodes an interpretation, redundant conditional statements are removed, as well as defeated statements. It ends with the rules for the `assume` phase as in Section 2.2.

In between, there are two CHriSM rules for each rule of $\mathcal{A}$. Each rule $r \in R$:

$$s_1 \land \ldots \land s_n \Rightarrow c_1 \land \ldots \land c_m \text{ assuming } a_1 \land \ldots \land a_k$$

is transformed into one simple probabilistic CHriSM rule:

$$P(r) \begin{array}{c} \Rightarrow \text{selected}(r) \end{array}$$

and one non-probabilistic CHriSM rule:

1 \begin{array}{c} \Rightarrow \text{selected}(r), \text{accept}(s_1, c_1), \ldots, \text{accept}(s_n, c_n) \end{array} \Rightarrow \text{append}([c_1, \ldots, c_m], [a_1, \ldots, a_k], NC), \text{accept}(c_1, NC), \ldots, \text{accept}(c_m, NC).

where all literals of the form $\neg x$ are encoded as `not(x)`.

**Correctness.** It is relatively straightforward to see that when the `assume` phase starts, `accept/2` encodes an interpretation that satisfies the criterion of Def. 5 w.r.t. the rule selection encoded by `selected/1`. It also does not contain gratuitous statements. However, the interpretation is not necessarily minimal. The `assume` phase searches for minimal interpretations by nondeterministically relaxing the assumptions. Since the relaxed `accept/2` constraints will cause the transformed PAL rules to be revisited, the criterion of Def. 5 remains satisfied.

3.3 Relationship between PAL and Defeasible Logic

In defeasible logic [11], there are three kinds of rules: strict rules (denoted with $\rightarrow$), defeasible rules (denoted with $\Rightarrow$), and undercutting defeaters (denoted with $\leadsto$). There is also a precedence relation $\prec$ over the non-strict rules, to settle conflicting rules. Conflicts are defined using a set of conflict sets, which should contain at least all sets of the form $\{\phi, \neg \phi\}$.

Without loss of generality (cf. Theorem 6 of [12]) we can assume that the precedence relation is empty, there are no undercutting defeaters, and the conflict set are just the atomic formulas and their negations.

Given a closed defeasible theory $D$ with a set of initial facts $F$ and a set of rules $R_S \cup R_D$ (where $R_S$ are the strict rules and $R_D$ are the defeasible rules), we can translate it to a probabilistic argumentation logic $\text{PAL}(D) = (S, A, R, P)$ as follows:
– The set of statements $S$ is the set of atoms appearing in $F$, $R_S$ and $R_D$;
– The set of assumptions $A$ consists of fresh symbols $a_1, \ldots, a_{|R_D|}$;
– The probability function $P$ is the constant function 1;
– The rules $R$ are constructed as follows:
  • For every fact $f \in F$, we add a rule $\Rightarrow f$.
  • For every strict rule $(X \rightarrow Y) \in R_S$, we add a rule $X \Rightarrow Y$.
  • For every defeasible rule $(X \Rightarrow Y) \in R_D$, we add a rule $X \Rightarrow Y$ assuming $a_i$, where $i$ is the number corresponding to the defeasible rule.
  • If the $i$-th and the $j$-th defeasible rule are conflicting rules, that is, if rule $i$ is of the form $(X \Rightarrow \psi) \in R_D$ and rule $j$ is of the form $(Y \Rightarrow \neg \psi) \in R_D$, then we add the rules $\psi \Rightarrow \neg a_j$ and $\neg \psi \Rightarrow \neg a_i$.
  • If the $i$-th defeasible rule $(X \Rightarrow \psi) \in R_D$ conflicts with a strict rule $(Y \rightarrow \bar{\psi}) \in R_S$ (where $\bar{\psi}$ denotes the negation of the literal $\psi$), then we add the rule $\bar{\psi} \Rightarrow \neg a_i$.

To illustrate this translation, let us look at a trivial example: the defeasible rules $\Rightarrow p$ and $\Rightarrow \neg p$. Since these rules contradict one another, neither $p$ nor $\neg p$ can be inferred. The translated PAL rules are the following:

$$
\Rightarrow p \text{ assuming } a_1
$$
$$
\Rightarrow \neg p \text{ assuming } a_2
$$
$$
p \Rightarrow \neg a_2
$$
$$
\neg p \Rightarrow \neg a_1
$$

This PAL has two valid interpretations (w.r.t. all rules): {$(p, \emptyset), (\neg a_2)$} and {$(\neg p, \emptyset), (\neg a_1)$}. As desired, neither $p$ nor $\neg p$ are acceptable.

Conjecture 1 (Weak Completeness). The above translation is complete in the following sense: for all literals $p$ in $\text{Lit}_D$, we have that if $D \not\models p$ (p is defeasibly derivable from $D$), then $p$ is plausible in $\text{PAL}(D)$ (w.r.t. the selection of all rules) and if $D \models \neg p$ (p is defeasibly refutable in $D$), then $p$ is not acceptable in $\text{PAL}(D)$.

Conjecture 2 (Soundness). The above translation is sound in the following sense: if the probability of $p$ in $\text{PAL}(D)$ is 1, then $D \models p$ (p is defeasibly entailed by $D$), and if the probability of $p$ in $\text{PAL}(D)$ is 0, then $D \not\models p$ (p is not defeasibly entailed by $D$).

4 Learning

For now, we have assumed the probabilities to be known in advance. As mentioned in the conclusion of [1], an issue is: where do these numbers come from? They suggest a statistical analysis of known precedents. The CHRiSM framework gives us exactly the tools needed to do this.

Instead of using fixed probabilities, we can make some or all probabilities learnable. We can then use a “training set” consisting of the outcomes of earlier similar cases — we call these the observations — to find a maximum likelihood probability distribution to fit the observations.
Table 1. Re-discovering the probabilities with CHiSM’s learning algorithm.

<table>
<thead>
<tr>
<th>Rule</th>
<th>Orig. prob.</th>
<th>Learned prob.</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>1</td>
<td>0.999999723</td>
</tr>
<tr>
<td>b</td>
<td>0.9</td>
<td>0.875267302</td>
</tr>
<tr>
<td>d</td>
<td>0.5</td>
<td>0.540243696</td>
</tr>
<tr>
<td>e</td>
<td>0.2</td>
<td>0.204545455</td>
</tr>
<tr>
<td>f</td>
<td>0.3</td>
<td>0.258644714</td>
</tr>
<tr>
<td>r1</td>
<td>1</td>
<td>0.999975929</td>
</tr>
<tr>
<td>r2</td>
<td>0.8</td>
<td>0.723429684</td>
</tr>
<tr>
<td>r3</td>
<td>0.5</td>
<td>0.547720330</td>
</tr>
<tr>
<td>r4</td>
<td>0.5</td>
<td>0.503165649</td>
</tr>
</tbody>
</table>

Some of the observations may be full observations, meaning that we not only know the final outcome, but also how exactly the reasoning went: what statements were put forward, what statements were accepted or rejected. More realistically, we only have partial observations: e.g. the final outcome is known, but not the intermediate steps. In CHiSM we can use both.

As a proof of concept, let us try to “rediscover” the original probabilities in our running example. Assume for the sake of the example we have a database of 1100 prior rulings, but we do not have the time and resources to read through all of them to find out exactly what the reasoning was. Say that we take 100 random samples and input them as full observations, for example

\[
\text{begin} \iff \text{accept}(\neg \text{app(r1)}, \emptyset), \text{accept}(d, \emptyset), \text{accept}(b, \emptyset), \text{accept}(a, \emptyset).
\]

In this example observation, even though both a and b were accepted, c was not accepted, either because rule r1 was not applied (remember, we do not know that it should have probability 1), or because its assumption \text{app(r1)} was refuted because d was accepted and rule r2 was applied.

The remaining 1000 cases are not studied in that much depth: all that was checked is who won the case (i.e. was c accepted or not?) and whether or not statement e (“the cow was mad”) was accepted. For the four possible combinations, the following counts were recorded:

- 62 times begin ==> accept(c,\emptyset), accept(e,\emptyset).
- 432 times begin ==> accept(c,\emptyset), ~accept(e,\emptyset).
- 138 times begin ==> ~accept(c,\emptyset), accept(e,\emptyset).
- 368 times begin ==> ~accept(c,\emptyset), ~accept(e,\emptyset).

In full observations, the right hand side of the large double arrow has to be exhaustive, so there is no need for explicit negation. In partial observations, the rhs is not an exhaustive enumeration, so explicit negation (denoted by tilde) can be useful, like in the above example.

The full observations above were obtained by simply taking 100 random samples (i.e. running the query \texttt{sample begin}) on the original program with explicit probabilities; the counts for the partial observations were obtained by computing the probabilities for each case multiplying them by 1000.

Now that we have a training set, we can use the built-in learning algorithm to find a probability distribution that fits the data. The resulting probabilities are shown in Table 1; they approximate the original probabilities reasonably well.
5 Conclusion

We defined probabilistic argumentation logic (PAL) and showed how it can be used for probabilistic legal reasoning in the style of [1]. We provided an implementation of PAL in CHRiSM through a straightforward encoding. Where previous work only showed by hand-performed examples how the problems could be solved, we provide a formalism and its encoding in CHRiSM that implements the problems and automates the generation of solutions. This automation goes much further than the original problem statement. The resulting CHRiSM program can be used to compute the probabilities of possible outcomes, to obtain random samples, and to learn some or all underlying probabilities, solving an open problem in [1].

It remains a topic for future work to relate PAL to other existing proposals for defeasible reasoning, argumentation, and non-monotonic logic in general. We have already indicated how Nute’s defeasible logic can be translated to a fragment of PAL in a sound and weakly complete way, although we postpone the proofs to future work.

References