October 2023

Precedent-based reasoning with incomplete cases

Daphne ODEKERKEN [a,b,1], Floris BEX [a,c] and Henry PRAKKEN [a]
[a] *Department of Information and Computing Sciences, Utrecht University*
[b] *National Police Lab AI, Netherlands Police*
[c] *Tilburg Institute for Law, Technology and Society, Tilburg University*

Abstract. We extend the result model for precedent-based reasoning with incomplete case bases. In contrast to regular case bases, these consist of incomplete cases for which not all dimension values need to be specified, but rather each dimension is assigned a set of possible values. The outcome of cases then applies for each (combination of) the possible dimension values. Building on earlier proposed notions of justification and stability for incomplete focus cases, we introduce the notion of possible justification statuses, which are required to maintain consistency of the incomplete case base. We demonstrate how these theoretic notions can be applied in practice for human-in-the-loop decision support, discuss their computational complexity and provide efficient algorithms.

Keywords. precedential constraint, incomplete cases

1. Introduction

Modelling reasoning with legal cases has been an important topic in the computational study of legal argument. This type of reasoning is applied to problems that are not decided by a legal rule, but by comparing the characteristics of cases. In particular, the characteristics of a new case for which the outcome still has to be decided (called *focus case*) are compared to the characteristics of precedent cases for which the outcome has already been assigned. These characteristics can be multi-valued ("*dimensions*") or boolean ("*factors*") – in this work, we consider factors to be a special case of dimensions. Each dimension has a direction, in the sense that some dimension values are more favourable to one of the outcomes than other dimension values.

In this paper, we focus on a particular research branch within the study of precedent-based reasoning, initiated by Horty in [1], that addresses the question how precedents constrain the possible decisions for a focus case. Horty provided multiple models, including a *result model* of precedential constraint. In a recent application, this model was used as a classifier for human-in-the-loop decision support [2,3]. Whereas most research on the result model presumes that the focus case is fully specified, that is not in the case in this application: for some dimensions, the values are either completely unknown or there are multiple possibilities. This motivated the introduction of the notion of stability in the

[1]Corresponding Author: Daphne Odekerken, d.odekerken@uu.nl.

context of incomplete focus cases in [3], where the incomplete focus case is called stable if for every choice of the possible dimension values the constrained outcome would be the same. If the outcome is stable towards one of the two sides (which we call PRO and CONTRA), then further refinement of the incomplete focus case is not necessary to make a decision. Otherwise, the analyst using the system can choose to either refine the incomplete focus case by reducing the possible values of some dimension(s) or make a decision and add this to the case base.

As far as we are aware, in all earlier work on the result model, including [3], the case base only allows "complete" cases, in the sense that there is exactly one value for each dimension. Consequently, whenever the analyst decides to add an incomplete focus case to the case base, still some choice should be made for these values, which would require either further investigation or a guess. Since neither option seems very desirable in general, we present in this paper an alternative approach that allows for incomplete information in not only the focus case, but also the case base. In order to do so, we introduce the notion of incomplete case bases in Section 3. Subsequently, we discuss how the justification and stability statuses, proposed for complete case bases in [3], can be computed efficiently with respect to an incomplete case base (Section 4). We then introduce in Section 5 the additional notion of possible justification statuses, which is required to maintain consistency of the incomplete case base used in the decision support system and conclude in Section 6. All proposed algorithms are added to the Python package for precedent-based reasoning at `https://git.science.uu.nl/D.Odekerken/lcbr`.

2. Preliminaries

First we recall the definitions of domain, fact situations, cases and the notion of precedential constraint from [4]. The *domain* $\mathscr{D}$ is the set of all dimensions related to the cases about which we want to reason, where a dimension is a tuple $d = (V, \leq_{\texttt{PRO}}, \leq_{\texttt{CONTRA}})$ such that V is a set of values that can be assigned to d and $\leq_{\texttt{PRO}}$ and $\leq_{\texttt{CONTRA}}$ are two partial orders on V such that $v \leq_{\texttt{PRO}} v'$ iff $v' \leq_{\texttt{CONTRA}} v$. The notation $V(\mathscr{D}, d)$ refers to the set of values that can be assigned to dimension d in domain $\mathscr{D}$. A *fact situation* c within a domain $\mathscr{D}$ is a set D of value assignments to all dimensions in $\mathscr{D}$. A value assignment is a pair (d, v) where $d \in \mathscr{D}$ and $v \in V(\mathscr{D}, d)$; note that fact situations require exactly one value for each dimension. The notation $v(d, c)$ denotes the value of dimension d in fact situation c. Based on the dimensions of the fact situation, an outcome o can be assigned, which can be either PRO or CONTRA. A fact situation c paired with the outcome o into a pair (c, o) is a *case*. We then refer to a set of cases as a *case base*.

Based on the ordering of dimension values, cases can be compared to each other, following [4, Definition 12]. The idea is that a case with dimensions having values that are "at least as PRO" as the values in case that is already assigned a PRO outcome is constrained to be PRO. Similarly, a case that is at least as CONTRA as a case with CONTRA outcome is assigned to be CONTRA. Using the cases from the case base, in specific situations the outcome of a new fact situation can be predicted ("forced") based on the so-called *a fortiori* constraint [1].

Definition 1 (A fortiori constraint)**.** *Let c_1 and c_2 be two fact situations within some domain $\mathscr{D}$ and o an outcome (PRO or CONTRA). Then c_1 is* at least as strong *as c_2 for the outcome o – written $c_1 \geq_o c_2$ – iff for all $d \in \mathscr{D}$: $v(d, c_1) \geq_o v(d, c_2)$. By the* a fortiori

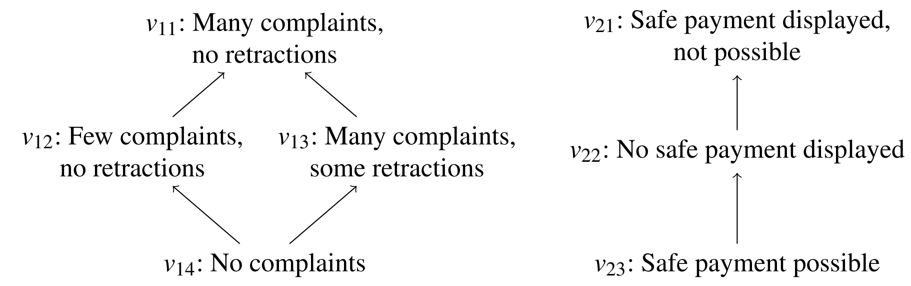
v_{11}: Many complaints, no retractions
v_{12}: Few complaints, no retractions
v_{13}: Many complaints, some retractions
v_{14}: No complaints
v_{21}: Safe payment displayed, not possible
v_{22}: No safe payment displayed
v_{23}: Safe payment possible

Figure 1. Hesse diagrams for the $\geq_{\mathtt{PRO}}$ ordering of the two dimensions in (a toy example version of) the web shop domain $\{d_1, d_2\}$. The dimension d_1, displayed on the left, has the four values $\{v_{11}, v_{12}, v_{13}, v_{14}\}$ while dimension d_2, on the right, has three values $\{v_{21}, v_{22}, v_{23}\}$. If there is a path from value a to b then $a \geq_{\mathtt{PRO}} b$.

constraint*, a fact situation c is* forced *towards outcome o (PRO or CONTRA) by a case base $\mathscr{C}$ iff there is some (c', o) in $\mathscr{C}$ such that $c \geq_o c'$.*

Note that a new case only has a forced outcome if there is a precedent in the case base for which the a fortiori constraint applies. The notion of justification directly uses the a fortiori constraint to assign an outcome to a focus case (i.e. any (new) fact situation that is not yet assigned an outcome) based on a *consistent* case base. The following definition for consistency is adapted from [5, Section III] for dimensions:

Definition 2 (Consistency of a case base)**.** *A case base $\mathscr{C}$ is* inconsistent *iff there are cases $(c_1, CONTRA)$ and (c_2, PRO) in $\mathscr{C}$ such that $c_1 \geq_{PRO} c_2$ and* consistent *otherwise.*

Definition 3 (Justification status)**.** *The* justification status *of a fact situation c given a consistent case base $\mathscr{C}$ is: PRO if c is forced towards outcome PRO by $\mathscr{C}$; CONTRA if c is forced towards outcome CONTRA by $\mathscr{C}$; and UNDECIDED otherwise.*

Example 1. *As a running example, we consider the domain $\mathscr{D}$ of classifying web shops, illustrated in Figure 1, for the decision whether a web shop is trustworthy (PRO) or not (CONTRA). $\mathscr{D}$ has two dimensions: $\mathscr{D} = \{d_1, d_2\}$. d_1 represents the number of complaints filed at the police, taking into account retracted complaints as well. It can be stated that web shops without complaints (v_{14}) at least as trustworthy (PRO) as, for example, web shops with many complaints and some retractions (v_{13}). Not all values of d_1 can be compared to each other: in the $\geq_{PRO}$ ordering illustrated in Figure 1, $v_{12} \not\geq_{PRO} v_{13}$ and $v_{13} \not\geq_{PRO} v_{12}$. d_2 represents the possibility of paying in a safe way. Whereas web shops that do not display this possibility (v_{22}) are in general less trustworthy than web shops that do have this possibility (v_{23}), it is even worse if a web shop (deceptively) displays the safe payment option while this is actually not possible (v_{21}). Suppose that a case base $\mathscr{C}$ consists of two cases $\mathscr{C} = \{(c_1, PRO), (c_2, CONTRA)\}$ where $c_1 = \{(d_1, v_{12}), (d_2, v_{23})\}$ and $c_2 = \{(d_1, v_{13}), (d_2, v_{22})\}$. Now consider a fact situation $c' = \{(d_1, v_{14}), (d_2, v_{23})\}$. Given that $c' \geq_{PRO} c_1$ and c_1 has the outcome PRO in $\mathscr{C}$, c' is forced towards outcome PRO by $\mathscr{C}$ and therefore has the justification status PRO.*

In practice, for new fact situations the exact value of each dimension is not immediately known [2]. Therefore, we would like to take multiple values into account for specific dimensions. An explicit representation of these possible values is enabled by the notion of incomplete fact situation, which was introduced in [3].

Definition 4 (Incomplete fact situation)**.** *An* incomplete fact situation *c within a domain $\mathcal{D}$ is the assignment of a possible value set for each dimension: for each $d = (V, \leq_{PRO}, \leq_{CONTRA})$ in $\mathcal{D}$ the incomplete fact situation contains a tuple (d, V_p) where $V_p \subseteq V(\mathcal{D}, d)$ is the set of possible values. The notation $V_p(d,c)$ denotes the set of possible dimension values of dimension d in incomplete fact situation c. For each $d \in \mathcal{D}$, $V_p(d,c)$ must contain at least one value.*

By obtaining more information on an incomplete fact situation, the set of possible values can be reduced. Once all these uncertainties are resolved, the fact situation is no longer incomplete. Completions of an incomplete fact situation are (complete) fact situations that are obtained by selecting a single value from the set of possible values for each dimension.

Definition 5 (Completions)**.** *A* completion *of an incomplete fact situation c is a fact situation c' such that for each $(d,v) \in c'$: $v \in V_p(d,c)$.*

The stability status, introduced in [3], can then be used to identify situations in which the justification is the same for all completions, so further investigation is not needed:

Definition 6 (Stability status)**.** *Let $\mathcal{C}$ be a consistent case base and c an incomplete fact situation. Then the* stability status *of c given $\mathcal{C}$ is Stable-PRO if the justification status of each completion c' of c given $\mathcal{C}$ is PRO; Stable-CONTRA if the justification status of each completion c' of c given $\mathcal{C}$ is CONTRA; Stable-UNDECIDED if the justification status of each completion c' of c given $\mathcal{C}$ is UNDECIDED; and Unstable otherwise.*

Example 2. *In the incomplete fact situation $c_3 = \{(d_1, \{v_{12}, v_{14}\}), (d_2, \{v_{23}\})\}$ in the web shop domain, we do not know the exact value of d_1 (i.e., the number of complaints and retractions). For each completion c' of c_3: $c' \geq_{PRO} c_1$; therefore, c_3 is Stable-PRO.*

In practice, there will be many situations like in Example 2, where a decision can be taken without full investigation of the incomplete focus case, given that it is Stable-PRO or Stable-CONTRA w.r.t. the case base. There will, however, also be many situations where this is not the case. Then the user can opt to do further investigation, refining the possible dimension values of the incomplete focus case. Such a refinement is a partial completion. In a partial completion, part of the information may still be uncertain, whereas in a completion the exact value for each of the dimensions is certain.

Definition 7 (Partial completion)**.** *A* partial completion *of an incomplete fact situation c is an incomplete fact situation c' such that for each $(d, V'_p) \in c'$: $V'_p \subseteq V_p(d,c)$.*

Example 3. *The incomplete fact situation $c_4 = \{(d_1, \{v_{11}, v_{12}, v_{13}\}), (d_2, \{v_{21}, v_{23}\})\}$ in the web shop domain is Unstable: completions can be PRO, UNDECIDED or CONTRA. An analyst can decide upon further investigation, for example by trying to make a payment using the safe option (analysing d_2). If the analyst succeeds, c_4 should be replaced by $c_5 = \{(d_1, \{v_{11}, v_{12}, v_{13}\}), (d_2, \{v_{23}\})\}$. Note that c_5 is a partial completion of c_4.*

3. Incomplete case bases

As we have seen in Example 3, incomplete fact situations that are unstable can be refined, which results in a partial completion that may or not be stable. Now suppose that the analyst decides on an outcome that would apply for each completion of the incomplete fact situation and wants to add this information to the case base. Let us assume, for example, that the analyst decides that having a safe payment option is a very strong sign that a web shop is trustworthy and that therefore each of c_5's completions should have the outcome PRO. If the case base can only contain "complete" cases, as defined in Section 2, then this information cannot be expressed in a single case. We therefore extend the definitions of cases and case bases to an incomplete variant.

Definition 8 (Incomplete case base)**.** *Within a domain $\mathscr{D}$, an* incomplete case *is a pair (c', o), where c' is an incomplete fact situation within the domain $\mathscr{D}$ and o is the assigned outcome of the case, which can be either PRO or CONTRA. An* incomplete case base *$\mathscr{C}$ within a domain $\mathscr{D}$ is a set of incomplete cases within $\mathscr{D}$.*

Although an incomplete case base can only contain incomplete cases, it is still possible to add a case for which the precise dimension values are known. This can be done by choosing a singleton for each set of possible dimension values. For example, in a domain with one dimension d_1 and a (complete) fact situation $\{(d_1, v_1)\}$ that we assign the outcome PRO, the "incomplete" case $(\{(d_1, \{v_1\})\}, \text{PRO})$ is added to the incomplete case base.

Since incomplete case bases consist of incomplete cases rather than cases, we need to redefine the notions of a fortiori constraint and consistency.

Definition 9 (A fortiori constraint with incomplete case base)**.** *A fact situation c is* forced *towards outcome o (PRO or CONTRA) by an incomplete case base $\mathscr{C}$ iff there is some incomplete case (c', o) in $\mathscr{C}$ such that c' has at least one completion c'' such that $c \geq_o c''$.*

Example 4. *Let $\mathscr{C}'$ be a case base in domain $\mathscr{D}$ that is the "incomplete version of" $\mathscr{C}$ from Example 1: $\mathscr{C}' = \{(c_1, PRO), (c_2, CONTRA)\}$ where $c_1 = \{(d_1, \{v_{12}\}), (d_2, \{v_{23}\})\}$ and $c_2 = \{(d_1, \{v_{13}\}), (d_2, \{v_{22}\})\}$. For $\mathscr{C}' \cup \{(c_5, PRO)\}$, the fact situation $c'' = \{(d_1, v_{13}), (d_2, v_{23})\}$ is forced towards outcome PRO because of $c_5 = \{(d_1, \{v_{11}, v_{12}, v_{13}\}), (d_2, \{v_{23}\})\}$: c_5 has a completion $c_5' = \{(d_1, v_{13}), (d_2, v_{23})\}$ and $c'' \geq_{PRO} c_5'$. Note that c_5 also has completions that are not "at least as PRO as" c'', for example: $c_5'' = \{(d_1, v_{11}), (d_2, v_{23})\} \not\geq_{PRO} c''$. This does not influence the a fortiori constraint on c'', as having c_5 in the incomplete case base with outcome PRO should be interpreted as: each of c_5's completions should have the outcome PRO. The exact value of d_1 does not matter here as the analyst considered the value assignment (d_2, v_{23}), i.e., the web shop having a safe payment option, to be decisive for having the PRO outcome.*

Note that Definition 9 is an incomplete counterpart of Definition 1. Analogously, we present an incomplete version of Definition 2 of consistency in Definition 10. In words, an incomplete case base is consistent if it is not possible to find a completion of a CONTRA case that is at least as PRO as a completion of a PRO case.

Definition 10 (Consistency of an incomplete case base)**.** *An incomplete case base $\mathscr{C}$ is* inconsistent *iff there are incomplete cases $(c_1, CONTRA)$ and (c_2, PRO) in $\mathscr{C}$ such that c_1 has a completion c_1' and c_2 has a completion c_2' for which $c_1' \geq_{PRO} c_2'$. $\mathscr{C}$ is* consistent *iff it is not inconsistent.*

Example 5. *$\mathscr{C}'$ from Example 4 is consistent. Now recall $c_4 = \{(d_1, \{v_{11}, v_{12}, v_{13}\}), (d_2, \{v_{21}, v_{23}\})\}$; $\mathscr{C}' \cup \{(c_4, PRO)\}$ is inconsistent, just like $\mathscr{C}' \cup \{(c_4, CONTRA)\}$.*

Then justification and stability w.r.t. incomplete case bases are defined as in Definitions 3 and 6, except that the consistent case base is replaced by an incomplete one.

Example 6. *For the consistent incomplete case base $\mathscr{C}' \cup \{(c_5, PRO)\}$ in the web shop domain, the incomplete fact situation $c_6 = \{(d_1, \{v_{12}, v_{13}\}), (d_2, \{v_{23}\})\}$ is Stable-PRO.*

Finally note that adding incomplete fact situations with a Stable-PRO or -CONTRA outcome to an incomplete case base can never make the resulting case base inconsistent:

Proposition 1. *Let $\mathscr{C}$ be a consistent incomplete case base and c an incomplete fact situation. $o \in \{PRO, CONTRA\}$: if c is Stable-o w.r.t. $\mathscr{C}$ then $\mathscr{C} \cup \{(c, o)\}$ is consistent.*

Proof. Suppose that c is Stable-PRO w.r.t. $\mathscr{C}$; then for each completion c' of c, there is some $(c^*, \text{PRO}) \in \mathscr{C}$ that has some completion $c^{*\prime}$ such that $c' \geq_{\text{PRO}} c^{*\prime}$. Given that $\mathscr{C}$ is consistent, there are no (c_1, CONTRA) and (c_2, PRO) in $\mathscr{C}$ such that c_1 has a completion c_1' and c_2 has a completion c_2' for which $c_1' \geq_{\text{PRO}} c_2'$. Thus, assuming towards a contradiction that $\mathscr{C} \cup \{(c, \text{PRO})\}$ is inconsistent, the inconsistency must have been introduced by adding (c, PRO): there must be some $(c^\star, \text{CONTRA})$ in $\mathscr{C}$ and some completion c' of c such that $c^\star$ has a completion $c^{\star\prime}$ and $c^{\star\prime} \geq_{\text{PRO}} c'$. But then, by transitivity of $\geq_{\text{PRO}}$, $c^{\star\prime} \geq_{\text{PRO}} c^{*\prime}$, which would imply that $\mathscr{C}$ was inconsistent. From this contradiction we derive that $\mathscr{C} \cup \{(c, \text{PRO})\}$ is consistent. □

4. Complexity and algorithms

If we literally follow Definition 9 in order to compute the justification status given an incomplete case base and a fact situation, the resulting algorithm would require an exponential number of steps, as the number of completions of a given incomplete fact situation can be exponential. Therefore, we propose an alternative, more efficient, algorithm in this section. This algorithm is based on the following proposition:

Proposition 2 (Reformulation of the a fortiori constraint for incomplete case bases)**.** *Let $\mathscr{C}$ be a consistent incomplete case base in domain $\mathscr{D}$ and c a fact situation. Then for outcome o (PRO or CONTRA): c is forced towards outcome o by $\mathscr{C}$ iff there is some incomplete (c', o) in $\mathscr{C}$ where c' is such that for each $d \in \mathscr{D}$ there is some $v' \in V_p(d, c')$ such that $v(d, c) \geq_o v'$.*

Proof. Let $\mathscr{C}$ be a consistent incomplete case base, let c be a fact situation and let o be an outcome in $\{\text{PRO}, \text{CONTRA}\}$. We proceed by proving both directions.

- Suppose that c is forced towards outcome o by $\mathscr{C}$. Then by Definition 9 there is some incomplete (c', o) in $\mathscr{C}$ such that c' has at least one completion c'' such that $c \geq_o c''$. Then by Definition 1 for all $d \in \mathscr{D}$: $v(d, c) \geq_o v(d, c'')$. Given that c'' is a completion of c', it must be that $v(d, c'') \in V_p(d, c')$. So for each $d \in \mathscr{D}$, there is some v' ($v' = v(d, c'')$) in $V_p(d, c')$ such that $v(d, c) \geq_o v'$.
- Suppose that there is some incomplete (c', o) in $\mathscr{C}$ where c' is such that for each $d \in \mathscr{D}$ there is some $v' \in V_p(d, c')$ such that $v(d, c) \geq_o v'$. Now construct the fact situation c'' by selecting for each dimension $d \in \mathscr{D}$ one value v' from $V_p(d, c')$ such that $v(d, c) \geq_o v'$. Given that, for each dimension $d \in \mathscr{D}$, $v(d, c'') \in V_p(d, c')$, by Definition 5, c'' is a completion of c'. In addition, $c \geq_o c''$ by Definition 1. We thus conclude that c is forced towards outcome o by $\mathscr{C}$ (Definition 9). □

We use Proposition 2 to design a polynomial algorithm for computing the justification status of fact situation c given a consistent incomplete case base $\mathscr{C}$. This algorithm works by iterating over $\mathscr{C}$, searching for an incomplete case $(c',o) \in \mathscr{C}$ such that for each dimension in the domain, there is at least one possible value v' of the incomplete fact situation such that $v(d,c) \geq_o v'$. If such an incomplete case is found, then by Proposition 2, the justification status is o; otherwise, it is UNDECIDED.

Example 7. *For the consistent incomplete case base $\mathscr{C}' \cup \{(c_5, PRO)\}$, the fact situation $c'' = \{(d_1,v_{13}),(d_2,v_{23})\}$ is forced towards outcome PRO because of c_5. Note that for each dimension, only one possible value in c_5 needs to be "at least as PRO" as the value in c''. In this example, $V_p(d_1,c_5) = \{v_{11},v_{12},v_{13}\}$. To derive the PRO status of c'' it suffices that $v(d_1,c'') = v_{13} \geq_{PRO} v_{13}$, even though $v_{11} \not\geq_{PRO} v_{13}$ and $v_{12} \not\geq_{PRO} v_{13}$.*

The algorithm requires a single run on the incomplete case base and for each of the incomplete cases it performs one comparison with the focus case for each possible dimension value. This implies that the algorithm is polynomial, therefore the problem of deciding PRO/CONTRA-justification can be computed in polynomial time (so is in P).

Proposition 3. *Deciding if an incomplete fact situation is PRO/CONTRA w.r.t. some incomplete case base is in P.*

Thus the justification problem in a setting with incomplete case bases is in the same complexity class as justification with complete case bases (cf. [3, Section 3.1]). Next, we show that the same applies for PRO/CONTRA-stability: it is still CoNP-complete.

Proposition 4. *Deciding if an incomplete fact situation is Stable-PRO/Stable-CONTRA w.r.t. some incomplete case base is CoNP-complete.*

Proof. The problem is CoNP-hard because it is CoNP-hard w.r.t. complete case bases (proven in [3, Proposition 6]). It is in CoNP as a negative instance can be verified in polynomial time, given a completion of the incomplete fact situation that is not PRO (or CONTRA) as a certificate, using the justification algorithm proposed in this section. □

To conclude this section, we provide an algorithm for stability with an incomplete case base. This algorithm is similar to the algorithm from [3] for complete case bases. The idea of the algorithm is to first construct the "most PRO" and "most CONTRA" completions of the incomplete fact situation. After that, the justification algorithm is applied to each of these completions. If all have the same justification status, the incomplete fact situation was stable. These extreme completions are defined as in [3, Definition 14]:

Definition 11 (Extreme completions)**.** *Let c be an incomplete fact situation within a domain $\mathscr{D}$ and let o be an outcome (PRO or CONTRA). For each of the dimensions $d_i \in \mathscr{D}$, let the most o values of this dimension in c be $\{V'_p \subseteq V_p(d,c) \mid$ for each $v'_p \in V'_p$: for each $v_p \in V_p$: if $v_p \geq_o v'_p$ then $v'_p \geq_o v_p\}$. The set of most o completions of c is $\{(d_i,v_i) \mid d_i \in \mathscr{D}, v_i \in$ the most o values of dimension d_i in $c\}$.*

Example 8. *Consider again the consistent incomplete case base $\mathscr{C}' \cup \{(c_5, PRO)\}$ in the web shop domain $\mathscr{D}$. The incomplete fact situation $c_6 = \{(d_1,\{v_{12},v_{13}\}),(d_2,\{v_{23}\})\}$ has two most CONTRA completions: $\{(d_1,v_{12}),(d_2,v_{23})\}$ and $\{(d_1,v_{13}),(d_2,v_{23})\}$. The first is forced to be PRO by c_1; the second is forced to be PRO as well, by c_5. Since all most CONTRA completions are PRO, c_6 must be Stable-PRO w.r.t. $\mathscr{C}' \cup \{(c_5, PRO)\}$.*

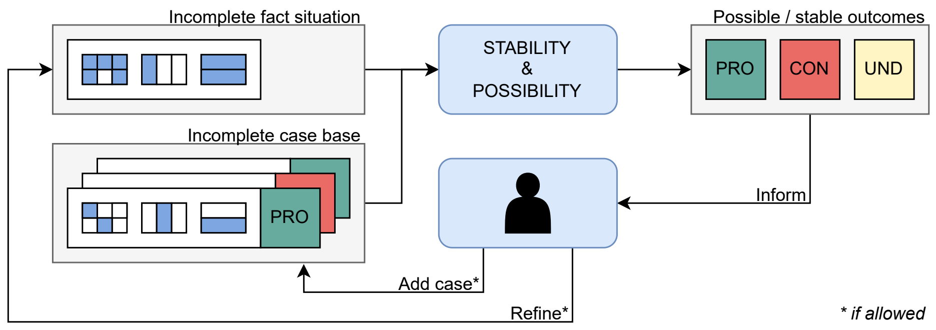

Figure 2. Illustration of the human-in-the-loop decision-making process, where the stability algorithm (Section 4) is used to check if the incomplete fact situation has a forced outcome given its incomplete case base. If not, the user chooses from the allowed actions (as specified in Section 5), which could include refining the incomplete fact situation or assigning an outcome and adding the resulting incomplete case to the incomplete case base.

5. Maintaining consistency

In Section 4 we assumed that the justification and stability algorithms are applied to an incomplete case base that is consistent, in accordance with Definition 10. In the human-in-the-loop decision support system, illustrated in Figure 2, we can guarantee consistency by posing restrictions on the incomplete fact situations that are allowed to be added to the case base. In the original version, where only complete fact situations could be added to the case base, it sufficed to check that cases with justification status PRO were not added with a CONTRA outcome and vice versa [3]. This does not apply to the extension presented in this paper, where *incomplete* fact situations are added to the case base, since the notion of justification status is only defined for *complete* fact situations that are used as focus case. We therefore introduce the notion of *possible* justification statuses and show how they can be used to guarantee that the consistency of the incomplete case base is maintained after an adding an incomplete case.

Informally, the possibility status is the natural counterpart of the stability status: whereas an incomplete fact situation is Stable-PRO if the justification status of *each* of its completions is PRO, it is Possible-PRO if *some* of them is PRO.

Definition 12 (Possibility statuses)**.** *Given a consistent incomplete case base $\mathscr{C}$ and a justification status j (PRO, CONTRA or UNDECIDED), an incomplete fact situation c is* Possible-j *if the justification status of some completion c' of c given $\mathscr{C}$ is j.*

Note that an incomplete fact situation can be both Possible-PRO and Possible-CONTRA. In addition, it is also possible that it is neither Possible-PRO nor Possible-CONTRA: this is exactly the case if the incomplete fact situation is Stable-UNDECIDED. Finally, each incomplete fact situation should have at least one possibility status – so it should be Possible-PRO, -CONTRA or -UNDECIDED (or a combination) – since each incomplete fact situation has at least one completion. Next, we show how this new notion of possibility can be applied for verifying if the incomplete case base remains consistent after adding an incomplete fact situation with a specific outcome.

P-PRO	**P-CONTRA**	**P-UNDEC**	**Stability**	**Add PRO**	**Add CONTRA**	**Refine**
Yes	No	No	S-PRO	Obsolete	Forbidden	Obsolete
No	Yes	No	S-CONTRA	Forbidden	Obsolete	Obsolete
No	No	Yes	S-UNDEC	Allowed	Allowed	Allowed
Yes	Yes	No	Unstable	Forbidden	Forbidden	Required
Yes	No	Yes	Unstable	Allowed	Forbidden	Allowed
No	Yes	Yes	Unstable	Forbidden	Allowed	Allowed
Yes	Yes	Yes	Unstable	Forbidden	Forbidden	Required

Table 1. Possibility statuses, their relation to stability and the allowed actions for the user. In this table, P is short for Possible, S is short for Stable and UNDEC is short for UNDECIDED.

Proposition 5 (Maintaining consistency)**.** *Let $\mathscr{C}$ be an incomplete case base that is consistent and let c be an incomplete fact situation.*

1. *$\mathscr{C} \cup (c, PRO)$ is consistent iff c is **not** Possible-CONTRA given $\mathscr{C}$; and*
2. *$\mathscr{C} \cup (c, CONTRA)$ is consistent iff c is **not** Possible-PRO given $\mathscr{C}$.*

In Table 1 we summarise the relation between each combination of possibility statuses, the corresponding stability status and the actions that the user can or should take (i.e., refine the case or add it to the case base with some outcome). For completeness, we also include the Possible-UNDECIDED status, although it does not play a role in maintaining consistency. Actions that introduce an inconsistency according to Proposition 5 are forbidden; all other actions are allowed. For incomplete fact situations that are both Possible-PRO and Possible-CONTRA, the only allowed action is refinement. Therefore, we marked this action as required. Finally, if an incomplete fact situation c is Stable-o (o being PRO or CONTRA) then adding (c,o) to the incomplete case base $\mathscr{C}$ is obsolete, in the sense that for any new fact situation c', the justification status of c' given $\mathscr{C} \cup \{(c,o)\}$ is exactly the same as its justification status given $\mathscr{C}$. Refinement does not have any effect either, as every partial completion of c will be Stable-o as well.

Given that the Possible-PRO and Possible-CONTRA statuses are required for deciding if an incomplete focus case with a specific outcome can be added to the incomplete case base, we need to compute these possibility statuses in an efficient way. Fortunately, this can be done in polynomial time, using Proposition 6.

Proposition 6 (Reformulation of possibility status)**.** *Let $\mathscr{C}$ be a consistent incomplete case base and c an incomplete fact situation and o an outcome (PRO or CONTRA). Then c is Possible-o given $\mathscr{C}$ iff there is some incomplete (c', o) in $\mathscr{C}$ where c' is such that for each $d \in \mathscr{D}$ there are some $v' \in V_p(d, c')$ and $v \in V_p(d, c)$ such that $v \geq_o v'$.*

Proposition 6 can be used in polynomial algorithms for finding the Possible-PRO and Possible-CONTRA statuses in the following way: given an incomplete case base $\mathscr{C}$ and incomplete fact situation c, iterate over all incomplete cases in $\mathscr{C}$. If we find come incomplete case $c' \in \mathscr{C}$ for which in all dimensions there is some $v' \in V_p(d, c')$ such that there is some $v \in V_p(d, c)$ such that $v \geq_{\mathsf{PRO}} v'$ then we conclude that c is Possible-PRO given $\mathscr{C}$, based on Proposition 6. The algorithm for Possible-CONTRA is analogous. Using these algorithms, the consistency check can be performed in polynomial time:

Proposition 7. *For any consistent incomplete case base $\mathscr{C}$, incomplete fact situation c and outcome o (PRO or CONTRA), it can be decided in polynomial time whether $\mathscr{C} \cup \{(c, PRO)\}$ or $\mathscr{C} \cup \{(c, CONTRA)\}$ is inconsistent.*

Proof. For any consistent incomplete case base $\mathscr{C}$ and incomplete fact situation c, it can be decided in polynomial time whether c is Possible-PRO and/or c is Possible-CONTRA given $\mathscr{C}$ using the algorithms described above, which require only a single run on $\mathscr{C}$. Then by Proposition 5, $\mathscr{C} \cup (c, \text{PRO})$ is consistent iff c is not Possible-CONTRA and $\mathscr{C} \cup (c, \text{CONTRA})$ is consistent iff c is not Possible-PRO. □

6. Conclusion and future work

We have shown how the human-in-the-loop decision support system introduced in [3] can be extended such that it becomes possible to add cases to the case base while their fact situation is incomplete. This is a convenient feature in situations where the user is certain of a specific outcome, regardless of the remaining uncertainties concerning the dimension values. We have shown that this extension does not influence the complexity classes in which the justification and stability problems are situated and provided algorithms for solving these problems. The extension does have an impact on the check that is required to maintain consistency of the case base: to guarantee consistency, we proposed a polynomial algorithm based on the newly introduced notion of possibility statuses.

The idea of making a decision based on part of the characteristics of the precedent seems to be strongly related to the reason model, for which various definitions have been proposed [4,6,7,8]. In future work, we plan to study how these definitions can be expressed in terms of incomplete fact situations and case bases. In addition, we aim to investigate two yet untouched aspects of the incomplete case base extension proposed here. First, we would like to study the complexity of the relevance problem, introduced in [3] for a setting with complete case bases, when applied to incomplete case bases. The second aspect is inconsistency handling. There may be situations in which the user is determined to add the incomplete focus case c with a specific outcome o to the case base, even though this is inconsistent with earlier decisions. It would then be convenient to have an automatic procedure for revision of the case base that adapts the case base in some minimal way, enabling the addition of (c, o) without inconsistencies.

References

[1] Horty J. Rules and reasons in the theory of precedent. Legal Theory. 2011;17:1-33.

[2] Odekerken D, Bex F. Towards transparent human-in-the-loop classification of fraudulent web shops. In: Villata S, Harasta J, Kremen P, editors. Proceedings of the 33rd International Conference on Legal Knowledge and Information Systems. IOS Press; 2020. p. 239-42.

[3] Odekerken D, Bex F, Prakken H. Justification, stability and relevance for case-based reasoning with incomplete focus cases. In: Proceedings of the 19th International Conference on Artificial Intelligence and Law; 2023. p. 177-86.

[4] Horty J. Reasoning with dimensions and magnitudes. Artificial Intelligence and Law. 2019;27:309-45.

[5] Horty J. The result model of precedent. Legal Theory. 2004;10(1):19-31.

[6] Horty J. Modifying the reason model. Artificial Intelligence and Law. 2021;29:271-85.

[7] Prakken H. A formal analysis of some factor-and precedent-based accounts of precedential constraint. Artificial Intelligence and Law. 2021;29(4):559-85.

[8] Rigoni A. Representing dimensions within the reason model of precedent. Artificial Intelligence and Law. 2018;26:1-22.