Statutory Interpretation by Case-based Reasoning through Abductive Logic Programming

Ken Satoh
Division of Electronics and Information Enginnering, Hokkaido University
ksatoh@db.huee.hokudai.ac.jp

Abstract

This paper presents a method of solving statutory interpretation problem in legal domain by using abductive logic programming. To interpret a rule, we need to make an interpretation for legal predicates and terms in the rule and this interpretation is subject to change according to context. In this paper, we solve this problem by comparing and contrasting cases like CABARET system [Rissland and Skalak, 1991]. This is done by translating case-based reasoning into abductive logic programming so that similarity between cases w.r.t. every undefined proposition in the rule is regarded as a set of abducibles which can be changed according to context. Our method is sound and complete in the sense that every possible interpretation is exactly computed.

1 Introduction

The *qualification problem* [McCarthy, 1980] is a serious problem in representing commonsense knowledge. For example, in order to represent the appropriate conditions for an action in real world, an enormous number of qualifications would be necessary since in real world, there are many exceptions which prevent the action from a successful result.

Even legal domain is not an exception for the problem. At the time when a rule is written, we cannot foresee every situation for the application of the rule. Suppose that we rigorously define the applicable condition of the rule so that no exception can be allowed. Even if the application of the rule in the current situation violates social welfare, we cannot avoid the application of the rule because of the representation of condition. Symmetrically, even if the application of the rule in the current situation is desirable in the spirit of the rule, we cannot apply the rule if the conditions do not match with the situation.

From the point of view of knowledge representation, it is interesting to see how law makers solve the 'qualification problem' in legal domain. One possible solution is to rewrite rules whenever an exception arises, but rewriting rules usually takes much time. In stead of rewriting, they (sometimes contrary to their intention) leave some room for interpretation of terms or predicates in rules so that the rule can be applied to unpredicted exceptional situations by changing their interpretation.

This solution, however, makes it difficult to apply a rule straightforwardly for the current situation because of the ambiguity of definition of terms or predicates. The problem of deciding whether a rule is applicable to the current situation or not by deciding the meaning of such ambiguous terms and predicates in the rule is called *statutory interpretation problem*. To solve the problem, "one compares and contrasts precedents with the instant case, reasons about the similarities and differences with past cases, and ultimately argues or explains why a previous interpretation can (or cannot) be applied" [Rissland and Skalak, 1991, page 843]. This reasoning can be regarded as a 'learning process' of definition of terms and predicates by using cases. In legal reasoning, to make matters more complicate, the interpretation is changed according to the position (for example, plaintiff and defendant).

CABARET system [Rissland and Skalak, 1991] is the first to consider the statutory interpretation problem using case-based reasoning in AI. In CABARET system, the interpretation is decided by similar cases relevant to the current case and the interpretation is changed according to one's point of view by combining rule-based reasoning and case-based reasoning.

In this paper, we restrict ourselves to concerning about the interpretation of propositions and propose a method of implementing such case-based reasoning mechanism for statutory interpretation through abductive logic programming.

We extend our method in [Satoh, 1996] which implements HYPO-style system [Ashley, 1990; Ashley and Aleven, 1994] by translating cases into abductive logic programming. In our previous method, we only use a casebase and decides which factors are relevant in order to make the current case favorable for one's position and refers to only one case for the current case. In this paper, we use not only a casebase, but also a rule base for the statutory interpretation and decide which factors are relevant for multiple propositions in the condition of the rule simultaneously and refer multiple cases to construct

an interpretation for the rule.

The semantics of abductive logic programming is based on generalized stable model [Kakas and Mancarella, 1990] reflecting a dynamic aspect of abducibles according to goals. We translate a casebase and a rule base into an abductive logic program and the position into its goal and show a correspondence between abducibles in a generalized stable model for a translated program and important factors for each proposition. This correspondence represents a dynamic change of similarity according to the position.

We use an existing procedure [Satoh and Iwayama, 1992] based on the semantics which is an enhanced procedure of [Eshghi and Kowalski, 1989] with bottom-up checking of integrity constraints. The procedure is sound and complete for a finitely grounded program in the sense that every generalized stable model is exactly computed. Since the translated program can be finitely grounded, we can guarantee that every possible interpretation of a rule is exactly computed thanks to the above correspondence of a model and important factors.

2 Formalization

We begin with definitions of a rule base and a casebase, then we define relevance criterion between cases to give an interpretation of propositions.

Definition 1 *Let $\mathcal{P}$ be a finite set of propositions. We divide $\mathcal{P}$ into two parts; $\mathcal{P}_{basic}$ called* a set of basic propositions *and $\mathcal{P}_{derived}$ called* a set of derived propositions.

Basic propositions are propositions for which we have to make an interpretation and derived propositions are defined by a rule using basic propositions.

We firstly define cases and factors which are used for confirmation and rejection of basic propositions for the current case.

Definition 2 *Let $\mathcal{F}$ be a finite set called* a set of factors. *A subset of $\mathcal{F}$ is called a case and a set of cases, CB, is called a* casebase.

For each $P \in \mathcal{P}_{basic}$, we take two disjoint subset of $\mathcal{F}$, AF_P and PF_P which are called absolute relevant factors *for P and* possible relevant factors *for P, respectively.*

For each $P \in \mathcal{P}_{basic}$, we divide CB into two disjoint set of cases, P^+ and P^-, which are called positive cases *for P and* negative cases *for P, respectively.*

A case is represented as a set of factors which are true in the case. If a factor belongs to the case, the factor is true and vice versa.

We consider a set of factors which will be used to interpret a proposition tentatively according to casebase. AF_P is a set of known critical factors to decide whether the proposition is true and PF_P is a set of factors that might be critical for the decision. It is not necessary that $AF_P \cup PF_P = \mathcal{F}$. If there is a factor which belongs to neither AF_P nor PF_P, this factor is regarded as absolutely irrelevant for the proposition.

Now, we consider a mechanism to decide whether the current case satisfies a basic proposition or not. The mechanism uses two kinds of cases; positive cases which are known to satisfy the proposition and negative cases which are known not to satisfy the proposition. Using these cases, we have a temporary interpretation of the proposition.

The current case is supposed to satisfy the proposition if the current case is similar to a positive case and not similar to any negative case. The similarity criterion is defined by status of focused factors of cases. If the status of every focused factors is same between cases, these cases are considered to be similar. This is done by comparing two intersections; intersection of a set of true factors in each case and a set of focused factors. If the intersections coincide, the status of every focused factors is same between cases.

Factors in AF_P are always focused whereas factors in PF_P can be either focused or focused according to the context. That is, focusing factors in PF_P is a source of dynamic change of similarity between cases.

Formally, we define the above as follows.

Definition 3 *Let C be a case. Then, we say that C* is positive w.r.t. a basic proposition P *if there exists a case $C_{P+} \in P^+$ and a subset of PF_P, IF_{P+} such that*

- $C \cap AF_P = C_{P+} \cap AF_P$, *and*
- $C \cap IF_{P+} = C_{P+} \cap IF_{P+}$, *and*
- *for every case $C_{P-} \in P^-$, either $C \cap AF_P \neq C_{P-} \cap AF_P$, or $C \cap IF_{P+} \neq C_{P-} \cap IF_{P+}$.*

We call C_{P+} a supporting case to prove P for C *and IF_{P+}* a set of possible important factors to prove P for C.

Example 1 *Let* $\mathcal{F} = \{f1, f2, f3\}$, $\mathcal{CB} = \{c1, c2, c3\}$,
$c1 = \{f1, f2\}$, $c2 = \{f1, f3\}$, $c3 = \{\}$,
$\mathcal{P} = \{p1, p2, p3\}$
$\mathcal{P}_{basic} = \{p1, p2\}$ *and* $\mathcal{P}_{derived} = \{p3\}$,
$AF_{p1} = \{f1\}$, $PF_{p1} = \{f2\}$,
$AF_{p2} = \{\}$, $PF_{p2} = \{f1, f2, f3\}$,
$p1^+ = \{c1, c2\}$, $p1^- = \{c3\}$,
$p2^+ = \{c2, c3\}$, $p2^- = \{c1\}$.

Suppose that the current case c is $\{f1\}$. Let $C_{p1+} = c2$ and $IF_{p1+} = \{f2\}$. Then,

- $c \cap AF_{p1} = c2 \cap AF_{p1} = \{f1\}$
- $c \cap IF_{p1+} = c2 \cap IF_{p1+} = \{\}$
- *for $c3 \in p1^-$, $c \cap AF_{p1} \neq c3 \cap AF_{p1} = \{\}$*

Therefore, c is positive w.r.t. $p1$. Note that if $C_{p1+} = c2$, there is the other set of possible important factors $\{\}$. Moreover, there is the other supporting case $c1$ and for $c1$ there is a set of possible important factors $\{\}$.

Similarly, Let $C_{p2+} = c3$ and $IF_{p2+} = \{f2, f3\}$. Then,

- $c \cap AF_{p2} = c3 \cap AF_{p2} = \{\}$
- $c \cap IF_{p2+} = c3 \cap IF_{p2+} = \{\}$
- *for $c1 \in p2^-$, $c \cap IF_{p2+} \neq c1 \cap IF_{p2+} = \{f2\}$*

Therefore, c is positive w.r.t. p2. Note that if $C_{p2+} = c3$, *there is the other set of possible important factors* $\{f2\}$. *Moreover, there is the other supporting case c2 and for c2 there are two sets of possible important factors* $\{f2\}$ *and* $\{f1, f2\}$.

In the above example, to show that $c2$ is a supporting case to prove $p1$ for c, the following are shown:

- for the proposition $f1 \in AF_{p1}$, both of c and $c2$ satisfy $f1$ and for the proposition $f2 \in IF_{p1+}$, neither c nor $c2$ satisfies $f2$. Therefore, c is similar to $c2$.
- for the case $c3 \in p1^-$, $f1 \in AF_{p1}$ is not satisfied by $c3$ whereas $f1$ is satisfied by c. Therefore, c is not similar to $c3$.

Thus, there is a similar positive case to c and there is no similar negative case to c w.r.t. the similarity criterion, c is positive w.r.t. $p1$.

Symmetrically, we give a definition of negation of C w.r.t. a basic proposition as follows.

Definition 4 *Let C be a case. Then, we say that C* is negative w.r.t. a basic proposition P *if there exists a case* $C_{P^-} \in P^-$ *and a subset of* PF_P, IF_{P^-} *such that*

- $C \cap AF_P = C_{P^-} \cap AF_P$, *and*
- $C \cap IF_{P^-} = C_{P^-} \cap IF_{P^-}$, *and*
- *for every case* $C_{P+} \in P^+$, *either* $C \cap AF_P \neq C_{P+} \cap AF_P$, *or* $C \cap IF_{P^-} \neq C_{P+} \cap IF_{P^-}$.

We call C_{P^-} *a* supporting case to disprove P for C *and* IF_{P^-} a set of possible important factors to disprove P for C.

Example 2 *Consider the same setting as Example 1. Let* $C_{p2^-} = c1$ *and* $IF_{p2^-} = \{f1, f3\}$. *Then,*

- $c \cap AF_{p2} = c1 \cap AF_{p2} = \{\}$
- $c \cap IF_{p2^-} = c1 \cap IF_{p2^-} = \{f1\}$
- *for* $c2 \in p2^+$, $c \cap IF_{p2^-} \neq c2 \cap IF_{p2^-} = \{f1, f3\}$ *and for* $c3 \in p2^+$, $c \cap IF_{p2^-} \neq c1 \cap IF_{p2^-} = \{\}$

Therefore, c is negative w.r.t. p2.

Note that this is the only pair of a supporting case and a set of possible important factors that makes c to be negative w.r.t. p2.

In the above two examples, the current case can be either positive and negative w.r.t. a basic proposition $p2$ by changing a supporting case and possible important factors. So, these examples demonstrate that the above definition reflects dynamic change of similarity according to the position.

Now, we define confirmation and rejection of derived propositions. This is done by relating basic propositions with derived propositions by using a rule of the following form.

Definition 5 *A* rule *is a formula of the form*

$$B_1, ..., B_n \Rightarrow H (n \geq 1)$$

where $H \in \mathcal{P}_{derived}$ *and* $B_i \in \mathcal{P}_{basic} (n \leq i \leq 1)$. *We call H the conclusion of the rule and* B_i*'s the condition of the rule. We call a set of rules RB a* rule base. *The set of all rules with P in its conclusion are called* rules of P.

The above rule means that if $B_1, ..., B_n$ are satisfied then H is also satisfied. Reader might think that this definition is too restricted since there are no relation between derived propositions, but if the rules does not have recursive definition then we can translate them into a set of rules of the above form. We will return this issue later.

Definition 6 *Let C be a case. We say that C* is positive w.r.t. a derived proposition P *if there exists a rule R in the rules of P such that for every basic proposition B in the condition of R, C is positive w.r.t. B. We call R a* supporting rule to prove P for C.

We say that C is negative w.r.t. a derived proposition P *if for every rule R in the rules of P, there exists a basic proposition B in the condition of R such that C is negative w.r.t. B.*

To show that the current case is positive w.r.t. a derived proposition P, it is sufficient to find a rule of P such that the current case is positive w.r.t. every proposition in the condition of the rule. On the other hand, to show that the current case is negative w.r.t. a derived proposition, we must show that the current case is negative w.r.t. some basic proposition in the condition of every rule of P.

Example 3 *Suppose that there is only one rule of* $p3$, $p1, p2 \Rightarrow p3$ *in the situation of Example 1.*

Because of the supporting cases and possible important factors to prove p1 and p2 for c in Example 1. Thus, c is positive w.r.t. all propositions (p1 and p2) in the condition of the rule of p3, c is positive w.r.t. p3 as well.

On the other hand, because of the supporting case to disprove p2 for c in Example 2, c is negative w.r.t. a proposition p2 in the condition of the rule of p3. So, c is negative w.r.t. p3 as well.

In the above example, the current case can be either positive and negative w.r.t. a derived proposition $p3$ by changing possible important factors and supporting case. So, context dependent similarity affects to proving/disproving not only basic propositions but also derived propositions.

3 Translation

Now, we give a translation of the above reasoning into abductive logic programming. The translation consists of four parts; translation of a casebase, translation of the current case, translation of a rule base and translation of the position (proving or disproving of a proposition).

Our translation is *local* w.r.t. cases in the sense that a case can be directly translated into a clause without any reference to the other cases. This is important to reduce complexity of translation since cases will be added frequently.

Definition 7 Translation of casebase *w.r.t. a basic proposition P is as follows. Let C be a case in* $\mathcal{CB}$. *Let* $AF_P = \{A_1, ..., A_k, A_{k+1}, ..., A_l\}$ *where* $A_1, ..., A_k$ *are true in C and* $A_{k+1}, ..., A_l$ *are false in C, Let* $PF_P =$

$\{F_1, ..., F_m, F_{m+1}, ..., F_n\}$ *where* $F_1, ..., F_m$ *are true in* C *and* $F_{m+1}, ..., F_n$ *are false in* C,
If $C \in P^+$,
$P(id(C)) \leftarrow$
$af(A_1, P), ..., af(A_k, P)$,
$naf(A_{k+1}, P), ..., naf(A_l, P)$,
$pf(F_1, P), ..., pf(F_m, P)$,
$npf(F_{m+1}, P), ..., npf(F_n, P)$.
where $id(C)$ *denotes an identification symbol for a case* C.
And if $C \in P^-$,
$n\_P(id(C)) \leftarrow$
$af(A_1, P), ..., af(A_k, P)$,
$naf(A_{k+1}, P), ..., naf(A_l, P)$,
$pf(F_1, P), ..., pf(F_m, P)$,
$npf(F_{m+1}, P), ..., npf(F_n, P)$.
We add the following clauses:
For $A_i \in AF_P (i = 1, ..., l)$,
$af(A_i, P) \leftarrow cf(A_i)$.
$naf(A_i, P) \leftarrow ncf(A_i)$.
For $F_i \in PF_P (i = 1, ..., n)$,
$pf(F_i, P) \leftarrow cf(F_i)$.
$pf(F_i, P) \leftarrow \sim imp^*(F_i, P)$.
$npf(F_i, P) \leftarrow ncf(F_i)$.
$npf(F_i, P) \leftarrow \sim imp^*(F_i, P)$.
where $\sim$ *means* negation as failure *and* $imp^*(F_i, P)$ *is abducibles.*
Finally, we add the integrity constraint to avoid contradiction:
$\bot \leftarrow P(X), n\_P(Y)$.
where $\bot$ *means contradiction and* X *and* Y *are variables.*

Firstly, for a basic proposition P, every case C is translated into a clause such that

- its conclusion is a consequence of the case w.r.t. P; if the case is positive w.r.t. P then the head becomes $P(id(C))$ whereas if it is negative w.r.t. P then the head becomes $n\_P(id(C))$.
- its body consists of the status of the absolute/possible relevant factors in the case.

If the current case coincides with a case with respect to status of the absolute relevant factors and the possible important factors w.r.t. a considered proposition, the current case is supposed to have the same consequence as the matched case.

$af(F, P) \leftarrow cf(F)$ and $naf(F, P) \leftarrow ncf(F)$ are used for matching the status of absolute relevant factors in a case and the current case since $cf(F)$ and $ncf(F)$ represent the status of factors in the current case (see below). $af(F, P)$ is true if and only if $cf(F)$ is true and $naf(F, P)$ is true if and only if $ncf(F)$ is true.

The following rules are used for deciding the possible important factors and matching the status of possible important factors between a case and the current case.
$pf(F, P) \leftarrow cf(F)$.
$pf(F, P) \leftarrow \sim imp^*(F, P)$.
$npf(F, P) \leftarrow ncf(F)$.
$npf(F, P) \leftarrow \sim imp^*(F, P)$.

Suppose that a factor F is not important w.r.t. P. This means that $\sim imp^*(F, P)$ is assumed. Then, $pf(F, P)$ and $npf(F, P)$ become automatically true. This corresponds with the behavior that the factor F is ignored to match the current case and a case in a casebase w.r.t. P. On the other hand, suppose that a factor F is important w.r.t. P. This means that $imp^*(F, P)$ is assumed. Then, $pf(F, P) \leftarrow cf(F)$ and $npf(F, P) \leftarrow ncf(F)$ are the only clauses for $pf(F, P)$ and $npf(F, P)$ respectively. Then, these clauses are used to check whether the status of F coincides for a case and the current case.

Definition 8 Translation for the current case C *is as follows. Let* $F_1, ..., F_k$ *be factors in* C *and* $F_{k+1}, ..., F_m$ *be factors not in* C.
$cf(F_1) \leftarrow$.
$\vdots$
$cf(F_k) \leftarrow$.
$ncf(F_{k+1}) \leftarrow$.
$\vdots$
$ncf(F_m) \leftarrow$.

Definition 9 Translation of rule base *w.r.t. a derived proposition* P *is as follows. Let rules of* P *be as follows.*
$B_{11}, B_{12}, ..., B_{1n_1} \Rightarrow P$
$\vdots$
$B_{m1}, B_{m2}, ..., B_{mn_m} \Rightarrow P$
We add the following clauses for showing the current case to be positive w.r.t. P.
For every i*-th rule of* P $(i = 1, .., m)$,
$P([X_1, ..., X_{n_i}]) \leftarrow B_{i1}(X_1), ..., B_{in_i}(X_{n_i})$.
where $X_1, ..., X_{n_i}$ *are variables.*
And we add the following clause for showing the current case to be negative w.r.t. P.
For every combination of m *propositions such that each proposition* $B_k (k = 1, .., m)$ *in the combination is one in the condition of each rule of* P:
$n\_P([Y_1, ..., Y_m]) \leftarrow n\_B_1(Y_1), ..., n\_B_m(Y_m)$.
where $Y_1, ..., Y_m$ *are variables.*

By the above translation for disproving P, we check that there exists a disproved proposition in the condition of every rule of P.

Definition 10
To prove that c *is positive w.r.t.* P, *ask* $? - P(X)$ *to the program, and to prove that* c *is negative w.r.t.* P, *ask* $? - n\_P(X)$ *to the program where* X *is a variable.*

By asking the above query, we express our position whether we would like to prove/disprove the proposition.

Example 4 *The following is the translation of the situation in Example 1 with the rule of* $p3$, $p1, p2 \Rightarrow p3$.
$p1(c1) \leftarrow af(f1, p1), pf(f2, p1)$.
$p1(c2) \leftarrow af(f1, p1), npf(f2, p1)$.
$n\_p1(c3) \leftarrow naf(f1, p1), npf(f2, p1)$.
$af(f1, p1) \leftarrow cf(f1)$.
$naf(f1, p1) \leftarrow ncf(f1)$.
$pf(f2, p1) \leftarrow cf(f2)$.

$pf(f2,p1) \leftarrow \sim imp^*(f2,p1).$
$npf(f2,p1) \leftarrow ncf(f2).$
$npf(f2,p1) \leftarrow \sim imp^*(f2,p1).$
$\bot \leftarrow p1(X), n\_p1(Y).$
$n\_p2(c1) \leftarrow pf(f1,p2), pf(f2,p2), npf(f3,p2).$
$p2(c2) \leftarrow pf(f1,p2), npf(f2,p2), pf(f3,p2).$
$p2(c3) \leftarrow npf(f1,p2), npf(f2,p2), npf(f3,p2).$
For $F = f1, ..., f3,$
$pf(F,p2) \leftarrow cf(F).$
$pf(F,p2) \leftarrow \sim imp^*(F,p2).$
$npf(F,p2) \leftarrow ncf(F).$
$npf(F,p2) \leftarrow \sim imp^*(F,p2).$
$\bot \leftarrow p2(X), n\_p2(Y).$
$cf(f1) \leftarrow .$
$ncf(f2) \leftarrow .$
$ncf(f3) \leftarrow .$
$p3([X,Y]) \leftarrow p1(X), p2(Y).$
$n\_p3([X]) \leftarrow n\_p1(X).$
$n\_p3([X]) \leftarrow n\_p2(X).$

Semantics of the above translated programs is based on generalized stable model [Kakas and Mancarella, 1990] as follows.

Definition 11 *Let T be a logic program and Π_T be a set of ground clauses obtained by replacing all variables in each clause in T by every element of its Herbrand universe. Let M be a set of ground atoms from Π_T and Π_T^M be the following program.*

$$\Pi_T^M = \{H \leftarrow B_1, ..., B_k | \\ H \leftarrow B_1, ..., B_k, \sim A_1, ..., \sim A_m \in \Pi_T \\ \text{and } A_i \notin M \text{ for each } i = 1, ..., m.\}$$

Let $min(\Pi_T^M)$ be the least model of Π_T^M. A stable model *for a logic program T is M iff $M = min(\Pi_T^M)$ and $\bot \notin M$.*

Definition 12 *Let T be a logic program with abducible predicates and Θ be a set of ground instances of abducible predicates (called* abducibles*). A* generalized stable model *$M(\Theta)$ for T is a stable model of $T(\Theta)$ where $T(\Theta) = T \cup \{H \leftarrow | H \in \Theta\}$.*

The following theorem states that if there is a model for a translated program which satisfies a translated atom for a basic proposition, then the abducibles in the model correspond with important factors to prove the basic proposition for the current case.

Theorem 1 *Let P be a basic proposition. Suppose that there exists a generalized stable model $M(\Theta)$ for a translated program T such that $M(\Theta) \models P(id(C_{P+}))$ Then, C_{P+} is a supporting case to prove P for the current case and and $\{F | imp^*(F,P) \in \Theta\}$ is a set of possible important factors to prove P for the current case.*

The next theorem states that if there are important factors to prove a basic proposition for the current case, then there is a model which satisfies a translated atom for the basic proposition and contains abducibles corresponding with the important factors.

Theorem 2 *Let P be a basic proposition. Suppose C_{P+} is a supporting case to prove P for the current case and IF_{P+} is a set of possible important factors to prove P for the current case. Then, there exists a generalized stable model $M(\Theta)$ for a translated program T such that $M(\Theta) \models P(id(C_{P+}))$ and $\{imp^*(F,P) | F \in IF_{P+}\} \subset \Theta$.*

Similar theorems holds for disproving a basic predicate.

The following theorem states that there exists a rule to make the current case to be positive for a derived proposition if and only if there is a model for a translated program satisfying the translation of derived proposition.

Theorem 3 *Let P be a derived proposition. The following are equivalent.*

1. *There is a generalized stable model $M(\Theta)$ for T and cases $C_1, ..., C_n$ such that $M(\Theta) \models P([id(C_1), ..., id(C_n)])$.*
2. *There exists a supporting rule R to prove a derived proposition for the current case of the form:*
   $B_1, ..., B_n \Rightarrow P$
   such that each supporting case to prove B_i for the current case is C_i $(i = 1, ..., n)$.

Example 5 *Let T be a translated program in Example 4. Let*
$\Theta = \{imp^*(f2,p1), imp^*(f2,p2), imp^*(f3,p2)\}$ *and*
$M(\Theta) = \Theta \cup$
$\{cf(f1), ncf(f2), ncf(f3),$
$af(f1,p1), npf(f2,p1),$
$pf(f1,p2), npf(f1,p2), npf(f2,p2), npf(f3,p2),$
$p1(c2), p2(c3), p3([c2,c3])\}$
Then, $\Pi_{T(\Theta)}^{M(\Theta)}$ becomes $\Pi_{T(\Theta)}$ except that
$pf(f1,p2) \leftarrow.$
$npf(f1,p2) \leftarrow.$
instead of the following clauses:
$pf(F,P) \leftarrow \sim imp^*(F,P)$, *and*
$npf(F,P) \leftarrow \sim imp^*(F,P)$
where $(F,P) = (f2,p1), (f1,p2), (f2,p2), (f3,p2).$
Then, $min(\Pi_{T(\Theta)}^{M(\Theta)}) = M(\Theta).$
Therefore, $M(\Theta)$ is a generalized stable model for T. Thus, in this model, $M(\Theta) \models p1(c2)$. $c2$ corresponds with the supporting case C_{p1+} and abducible $imp^*(f2,p1)$ in Θ in this model corresponds with the possible important factors $IF_{p1+} = \{f2\}$ in Example 1. Similarly, $M(\Theta) \models p2(c3)$. $c3$ corresponds with the supporting case C_{p2+} and abducible $imp^*(f2,p2)$ and $imp^*(f3,p2)$ in Θ in this model correspond with the possible important factors $IF_{p2+} = \{f2, f3\}$ in Example 1.

We can show that there is a corresponding generalized stable model with every other pair of a supporting case and a set of possible important factors. This corresponds with the result of Example 1.

Moreover, in this model, $M(\Theta) \models p3([c2,c3])$ which corresponds with the result in Example 3.

The following theorem is for disproving a derived proposition.

Theorem 4 *Let P be a derived proposition. The following are equivalent.*

1. *There is a generalized stable model $M(\Theta)$ for T and cases $C_1, ..., C_m$ such that $M(\Theta) \models n\_P([id(C_1), ..., id(C_m)])$.*
2. *The current case is negative w.r.t. P and for every rule $R_i (i = 1, ..., m)$ of P, there exists a proposition B_i in the condition of R_i such that a case C_i is a supporting case to disprove B_i for the current case.*

Proofs are found in Appendix.

Example 6 *Let T be a translated program in Example 4. Let $\Theta = \{imp^*(f1, p2), imp^*(f3, p2)\}$ and*
$M(\Theta) = \Theta \cup$
$\{cf(f1), ncf(f2), ncf(f3),$
$af(f1, p1), pf(f2, p1), npf(f2, p1),$
$pf(f1, p2), pf(f2, p2), npf(f2, p2), npf(f3, p2),$
$p1(c1), p1(c2), n\_p2(c1), n\_p3([c1])\}$
Then, $\Pi_{T(\Theta)}^{M(\Theta)}$ becomes $\Pi_{T(\Theta)}$ except that
$pf(f2, p1) \leftarrow.$
$npf(f2, p1) \leftarrow.$
$pf(f2, p2) \leftarrow.$
$npf(f2, p2) \leftarrow.$
instead of the following clauses:
$pf(F, P) \leftarrow \sim imp^*(F, P)$, *and*
$npf(F, P) \leftarrow \sim imp^*(F, P)$
where $(F, P) = (f2, p1), (f1, p2), (f2, p2), (f3, p2)$.
Then, $min(\Pi_{T(\Theta)}^{M(\Theta)}) = M(\Theta)$.
Therefore, $M(\Theta)$ is a generalized stable model for T. Therefore, in this model, $M(\Theta) \models n\_p2(c1)$ and $c1$ corresponds with the supporting case C_{p2-} and a set of abducibles Θ in this model corresponds with the possible important factors $IF_{p2-} = \{f1, f3\}$ in Example 2. Note that we can show that this model is the only one which satisfies $n\_p2(c1)$ and this corresponds with the result of Example 2.
Moreover, in this model, $M(\Theta) \models n\_p3([c1])$ which corresponds with the result in Example 3.

This framework is implemented on a top-down proof procedure based on generalized stable models [Satoh and Iwayama, 1992]. For a program T whose Π_T is finite, this proof procedure is sound and complete in the sense that if there is a generalized stable model, then the proof procedure computes every abducibles which makes the query true and vice versa.

For a basic proposition, the procedure computes a supporting case as an answer of the variable in a translated query in Definition 10, and also computes possible important factors as abducibles. For a derived proposition, the procedure computes a list of supporting cases as an answer of the variable in the query and a union of possible important factors as abducibles. Since the above theorems guarantee the correspondence between every generalized stable model and every pair of supporting case and possible important factors, the proof procedure is guaranteed to compute every pair of supporting case and a set of possible important factors.

4 Legal Reasoning Example

We consider a simplified version of example in CISG (United Nations Convention on Contracts for the International Sale of Goods).

In the Article 25 of CISG,

> *A breach of contract committed by one of the parties is fundamental if it results in such detriment to the other party as substantially to deprive him of what he is entitled to expect under the contract, unless the party in breach did not foresee and a reasonable person of the same kind in the same circumstances would not have foreseen such a result.*

This is related to declaring the contract avoided. If the breach of contract is fundamental on a party, then the other party can declare the contract avoided. So, it is very important that the breach of contract is fundamental or not. But, the above statement is the only definition for a fundamental breach of contract in CISG and we need to decide whether the breach substantially deprive the other party of what he expects.

For simplicity, we omit "unless" part and read the above article of the rule form as:

$$dwe \Rightarrow bcf.$$

where bcf stands for "A breach of contract committed by one of the parties is fundamental" and dwe stands for "depriving the other party of what he is entitled to expect under the contract".

Suppose that we have the following cases in a casebase.

Case 1: The goods are delivered on time and there is no damage of goods. In this case, the seller does not deprive the buyer of what he expects.

Case 2: The goods are not delivered on time, and the buyer fixes an additional period of time, and the goods are delivered in the period. In this case, the seller does not deprive the buyer of what he expects, either.

Case 3: The goods are delivered on time, but the goods are out of order. However, the trouble can be repaired. So, the buyer fixes an additional period of time for repair and the repair is completed in the additional period. In this case, the seller does not deprive the buyer of what he expects, either.

Case 4: The goods are delivered on time, but the goods are out of order and the trouble cannot be repaired. In this case, the buyer is deprived of what he expects.

Case 5: The goods are not delivered on time and the buyer fixes an additional period of time, and the goods are not delivered in the period. In this case, the buyer is deprived of what he expects, too.

We will decide whether the following case satisfies bcf or not.

> The goods are delivered on time, but the goods are out of order. However, the trouble can be

repaired. So, the buyer fixes an additional period of time for repair, but the repair is not completed in the additional period.

Since the goods are remained to be out of order, it is desirable that the case satisfies bcf and therefore, the buyer should be able to declare the contract avoided. But, from the article, it is ambiguous whether the situation actually satisfies bcf. So, we decide it by finding similarity with one of the previous cases.

From the analysis of these cases, we consider the following factors.

dot: The goods are delivered on time.

fad: The buyer fixes an additional period for delivering the goods.

dia: The goods are delivered in the above additional period.

ooo: The goods are out of order.

rpl: The goods are repairable.

far: The buyer fixes an additional period for repair.

ria: The goods are repaired in the above additional period.

Therefore, in this example, we define the rule base and the casebase as follows.

1. $\mathcal{P}_{basic} = \{dwe\}$ and $\mathcal{P}_{derived} = \{bcf\}$.
2. $\mathcal{F} = \{dot, fad, dia, ooo, rpl, far, ria\}$.
3. $\mathcal{CB} = \{c1, c2, c3, c4, c5\}$ where
   $c1 = \{dot\}$
   $c2 = \{fac, dia\}$
   $c3 = \{dot, ooo, rpl, far, ria\}$
   $c4 = \{dot, ooo\}$
   $c5 = \{fac\}$
   Note that if the factor is not included in the case, the factor is false in the case.
4. $\mathcal{RB} = \{dwe \Rightarrow bcf\}$
5. $AF_{dwe} = \{\}$
   $PF_{dwe} = \{dot, fad, dia, ooo, rpl, far, ria\}$ [1]
6. $dwe^+ = \{c4, c5\}$ and $dwe^- = \{c1, c2, c3\}$
7. The current case $c = \{dot, ooo, rpl, far\}$

From the syntactical point of view, c seems to be similar to $c3$ since only the difference is ria. Therefore, from this similarity, we would conclude that the current case does not satisfy dwe, nor does bcf. Of course, this is one possible interpretation, but if we do want to conclude that the current case satisfies dwe, we have to find the similar case satisfying dwe by another similarity.

We translate the above into the following logic program:

$n\_dwe(c1) \leftarrow pf(dot, dwe),$
$\quad npf(fac, dwe), npf(dia, dwe),$
$\quad npf(ooo, dwe), npf(rpl, dwe),$
$\quad npf(far, dwe), npf(ria, dwe).$
$n\_dwe(c2) \leftarrow npf(dot, dwe),$
$\quad pf(fac, dwe), pf(dia, dwe),$
$\quad npf(ooo, dwe), npf(rpl, dwe),$
$\quad npf(far, dwe), npf(ria, dwe).$
$n\_dwe(c3) \leftarrow pf(dot, dwe),$
$\quad npf(fac, dwe), npf(dia, dwe),$
$\quad pf(ooo, dwe), pf(rpl, dwe),$
$\quad pf(far, dwe), pf(ria, dwe).$
$dwe(c4) \leftarrow pf(dot, dwe),$
$\quad npf(fac, dwe), npf(dia, dwe),$
$\quad pf(ooo, dwe), npf(rpl, dwe),$
$\quad npf(far, dwe), npf(ria, dwe).$
$dwe(c5) \leftarrow npf(dot, dwe),$
$\quad pf(fac, dwe), npf(dia, dwe),$
$\quad npf(ooo, dwe), npf(rpl, dwe),$
$\quad npf(far, dwe), npf(ria, dwe).$
For $F = dot, fad, dia, ooo, rpl, far, ria,$
$\quad pf(F, dwe) \leftarrow cf(F).$
$\quad pf(F, dwe) \leftarrow \sim imp^*(F, dwe).$
$\quad npf(F, dwe) \leftarrow ncf(F).$
$\quad npf(F, dwe) \leftarrow \sim imp^*(F, dwe).$
$\bot \leftarrow dwe(X), n\_dwe(Y).$
$cf(dot) \leftarrow .$
$ncf(fac) \leftarrow .$
$ncf(dia) \leftarrow .$
$cf(ooo) \leftarrow .$
$cf(rpl) \leftarrow .$
$cf(far) \leftarrow .$
$ncf(ria) \leftarrow .$
$bcf([X]) \leftarrow dwe(X).$
$n\_bcf([X]) \leftarrow n\_dwe(X).$

If we ask $? - bcf(X)$ to the above program, then we can get an answer $X = [c4]$ with abducibles $\{imp^*(ooo, dwe), imp^*(ria, dwe)\}$ as one of the possible interpretations.

Since $c4$ satisfies ooo and does not satisfy ria, we can read the above similarity as

> *If the goods are out of order and they are not repaired in the additional period of time for repair fixed by the buyer, the seller deprives the buyer of what the buyer expects.*

At a first glance, $c4$ does not seem to be related with the current case. But, from the translated program, we can show as an interpretation that it is similar to the current case because of the above similarity. After we showed this demonstration at a research meeting of AI and law in Japan, one professor in the law department pointed out that this similarity is based on the policy that "if the contractor suffers from the loss of benefit, the contract should be avoided". Before showing this similarity, no one ever pointed out this policy at the meeting. This would demonstrate that our proposed mechanism can help lawyers clear the policy behind the law which they unconsciously have and that the mechanism can be used as a supporting tool for a law decision.

[1] Since we do not know which factor is absolutely relevant, we set all the factors as possible relevant ones.

5 Extension

5.1 Acyclic Rule bases

If a rule base is acyclic, that is, every derived proposition does not refer to itself in its definition, we can unfold it into a "flat" structure of rules in Definition 5.

Example 7 *Consider the following rule base:*
$B_1, B_2 \Rightarrow P_1$
$B_3, B_4 \Rightarrow P_1$
$B_5, B_6 \Rightarrow P_2$
$B_7, B_8 \Rightarrow P_2$
$P_1, P_2, B_9 \Rightarrow P_3$
Then, we can translated the definition of P_3 *into the following "flat" version by unfolding each derived proposition to its body:*
$B_1, B_2, B_5, B_6, B_9 \Rightarrow P_3$
$B_3, B_4, B_5, B_6, B_9 \Rightarrow P_3$
$B_1, B_2, B_7, B_8, B_9 \Rightarrow P_3$
$B_3, B_4, B_7, B_8, B_9 \Rightarrow P_3$

5.2 Relevant Propositions

In this section, we discuss an extension of our framework on relevancy between basic propositions. If there is some relevancy between propositions and when different cases are independently retrieved for each proposition, it might lead to incoherency.

For example, suppose there is a rule $p1, p2 \Rightarrow p3$ and to prove $p3$, we must retrieve a case such that $p1$ and $p2$ are both true in the case. If we retrieve a case for $p1$ in which $p2$ is false and a different case for $p2$ in which $p1$ is false, the requirement of $p1$ and $p2$ for a retrieved case is violated.

This kind of mutual relevancy between propositions can be found in legal reasoning. For example, in paragraph (1) in the Article 14 of CISG[2],

> *A proposal for concluding a contract addressed to one or more specific persons constitutes an offer if it is sufficiently definite and indicates the intention of the offeror to be bound in case of acceptance. A proposal is sufficiently definite if it indicates the goods and expressly or implicitly fixes or makes provision for determining the quantity and the price.*

This paragraph says that in order for a proposal to be a contract, the proposal must show sufficient definiteness and offeror's intention. These propositions (sufficient definiteness and offeror's intention) are not mutually independent, but depend on each other. Even if the proposal shows offeror's intention, if there is no detail about proposal, then the proposal is doubtful to be a contract. On the other hand, even if the proposal does not show offeror's intention, if there is detailed information about proposal, then the proposal can be regarded as representing the offeror's intention. Therefore, in such cases, if we retrieve a similar case for each proposition independently, the retieval might hurt the overall coherency.

[2] I am very grateful to Prof. Sono for pointing out this example to me.

As a solution for the relevancy problem between propositions, we can introduce common variables between propositions in a translated program so that contradictory cases cannot be retrieved for relevant propositions.

Example 8 *We consider the same problem as Example 4 except that* $p1$ *and* $p2$ *are relevant. The program is same except the definition related with* $p3$*:*
$p3([X]) \leftarrow p1(X), p2(X)$
instead of $p3([X, Y]) \leftarrow p1(X), p2(Y)$.
Then, generalized stable models of this new program are restricted to such models in the old program that the supporting cases for $p1$ *and* $p2$ *coincide.*

6 Conclusion

In this paper, we provide a method of solving the statutory interpretation problem by using abductive logic programming in which we show correspondence between abducibles and possible important factors which gives an interpretation of undefined propositions in a rule.

The following are the future works.

- We would like to pursue other usage of this framework such as argument simulation in legal reasoning.
- We would like to extend factors to ones with some value ranges so that quantitative factors can be expressed.

References

[Ashley, 1990] Kevin D. Ashley. *Modeling Legal Argument: Reasoning with Cases and Hypotheticals*, MIT press, 1990.

[Ashley and Aleven, 1994] Kevin D. Ashley and Vincent Aleven. A Logical Representation for Relevance Criteria. S. Wess, K-D. Althoff and M. Richter (eds.), *Topics in Case-Based Reasoning, LNAI 837*, pages 338 – 352, 1994.

[Eshghi and Kowalski, 1989] Kave Eshghi and Robert A. Kowalski. Abduction Compared with Negation by Failure. In *Proceedings of the Sixth International Conference on Logic Programming*, pages 234 – 254, 1989.

[Kakas and Mancarella, 1990] Antonis Kakas and Paulo Mancarella. Generalized Stable Models: A Semantics for Abduction. In *Proceedings of the Ninth European Conference on Artificial Intelligence*, pages 385 – 391, 1990.

[McCarthy, 1980] John McCarthy. Circumscription—A Form of Non-Monotonic Reasoning, *Artificial Intelligence*, 13:27 – 39, 1980.

[Rissland and Skalak, 1991] Edwina L. Rissland and David Skalak. CABARET: Statutory Interpretation in a Hybrid Architecture. *International Journal of Man-Machine Studies*, 9(6):839 – 887, 1991.

[Satoh and Iwayama, 1992] Ken Satoh and Noboru Iwayama. A Query Evaluation Method for Abductive Logic Programming. In *Proceedings of the Joint International Conference and Symposium on Logic Programming*, pages 671 – 685, 1992.

[Satoh, 1996] Ken Satoh. Translating Case-Based Reasoning into Abductive Logic Programming. In *Proceedings of the Twelfth European Conference on Artificial Intelligence*, pages 142 – 146, 1996.

Appendix: Proofs of theorems

Proof of Theorem 1:
Let the clause for C_{P+} be

$P(c_{P+}) \leftarrow$
$af(A_1,P),...,af(A_k,P),$
$naf(A_{k+1},P),....,naf(A_l,P),$
$pf(F_1,P),...,pf(F_m,P),$
$npf(F_{m+1},P),....,npf(F_n,P),$

where $c_{P+} = id(C_{P+})$.
Since there is no other clause for C_{P+}, the body of this clause should be satisfied. Therefore, the following hold.

1. For each $af(A_i,P)(1 \leq i \leq k)$, $M(\Theta) \models af(A_i,P)$.
2. For each $naf(A_i,P)(k+1 \leq i \leq l)$, $M(\Theta) \models naf(A_i,P)$.
3. For each $pf(F_i,P)(1 \leq i \leq m)$, $M(\Theta) \models pf(F_i,P)$.
4. For each $npf(F_i,P)(m+1 \leq i \leq n)$, $M(\Theta) \models npf(F_i,P)$.

Then,

1. $M(\Theta) \models af(A_i,P)$ means that $M(\Theta) \models cf(A_i,P)$. Then, the following hold:
   - $af(A_i,P)$ in the body of the clause of $P(c_{P+})$ means $A_i \in C_{P+}$ and $A_i \in AF_P$.
   - $M(\Theta) \models cf(A_i,P)$ means $A_i \in C$ where C is the current case.

   Thus, $A_i \in C_{P+}$ and $A_i \in AF_P$ implies $A_i \in C$. This means that $C_{P+} \cap AF_P \subseteq C$, thus $C_{P+} \cap AF_P \subseteq C \cap AF_P$.
2. In a symmetrical way, $M(\Theta) \models npf(A_i,P)$ leads to $C \cap AF_P \subseteq C_{P+} \cap AF_P$. Therefore, $C \cap AF_P = C_{P+} \cap AF_P$.
3. $M(\Theta) \models pf(F_i,P)$ means that either $M(\Theta) \models cf(F_i,P)$ or $M(\Theta) \not\models imp^*(F_i,P)$. Then, the following hold:
   - $pf(F_i,P)$ in the body of the clause of $P(c_{P+})$ means $F_i \in C_{P+}$.
   - $M(\Theta) \models cf(F_i,P)$ means $F_i \in C$.
   - $M(\Theta) \not\models imp^*(F_i,P)$ means $F_i \notin IF_{P+}$ where $IF_{P+} = \{F_i | imp^*(F_i,P) \in \Theta\}$.

   Thus, $F_i \in C_{P+}$ and $F_i \in IF_{P+}$ implies $F_i \in C_{P+}$. This means that $C_{P+} \cap IF_{P+} \subseteq C$, thus $C_{P+} \cap IF_{P+} \subseteq C \cap IF_{P+}$.
4. In a symmetrical way, $M(\Theta) \models npf(F_i,P)$ leads to $C \cap IF_{P+} \subseteq C_{P+} \cap IF_{P+}$. Therefore, $C \cap IF_{P+} = C_{P+} \cap IF_{P+}$.

Since $M(\Theta)$ satisfies every integrity constraint, $\bot \leftarrow P(X), n\_P(Y)$, and we have already shown that there exists some X for $P(X)$, for every $C_{P-} \in P^-$, $M(\Theta) \not\models n\_P(c_{P-})$ where $c_{P-} = id(C_{P-})$. Let the clause for C_{P-} be

$n\_P(c_{P-}) \leftarrow$
$af(A_1,P),...,af(A_s,P),$
$naf(A_{s+1},P),....,naf(A_t,P),$
$pf(F_1,P),...,pf(F_u,P),$
$npf(F_{u+1},P),....,npf(F_v,P).$

Then, at least, one of the following hold.

1. There exists $af(A_i,P)(1 \leq i \leq s)$ such that $M(\Theta) \not\models af(A_i,P)$.
2. There exists $naf(A_i,P)(s+1 \leq i \leq t)$ such that $M(\Theta) \not\models naf(A_i,P)$.
3. There exists $pf(F_i,P)(1 \leq i \leq u)$ such that $M(\Theta) \not\models pf(F_i,P)$.
4. There exists $npf(F_i,P)(u+1 \leq i \leq v)$ such that $M(\Theta) \not\models npf(F_i,P)$.

Then,

1. Suppose that there exists $af(A_i,P)$ such that $M(\Theta) \not\models af(A_i,P)$. This means that $M(\Theta) \not\models cf(F_i,P)$. Therefore, $A_i \in C_{P-}$ and $A_i \in AF_P$ but $A_i \notin C$. This means $C \cap AF_P \neq C_{P-} \cap AF_P$.
2. Suppose that there exists $naf(A_i,P)$ such that $M(\Theta) \not\models naf(A_i,P)$. In a similar way, we can show that $C \cap AF_P \neq C_{P-} \cap AF_P$.
3. Suppose that there exists $pf(F_i,P)$ such that $M(\Theta) \not\models f(F_i,P)$. This means that $M(\Theta) \not\models cf(F_i,P)$ and $M(\Theta) \models imp^*(F_i,P)$. Therefore, $F_i \in C_{P-}$ and $F_i \in IF_{P+}$ but $F_i \notin C$. This means $C \cap IF_{P+} \neq C_{P-} \cap IF_{P+}$.
4. Suppose that there exists $npf(F_i,P)$ such that $M(\Theta) \not\models npf(F_i,P)$. In a similar way, we can show that $C \cap IF_{P+} \neq C_{P-} \cap IF_{P+}$.

Therefore, $C \cap AF_P \neq C_{P-} \cap AF_P$ or $C \cap IF_{P+} \neq C_{P-} \cap IF_{P+}$.

Therefore, C_{P+} is a supporting case and IF_{P+} is a set of important properties to prove P for the current case C. □

To prove Theorem 2, we need the following lemma.

Lemma 1 *Let C be a current case. Suppose that there is a case C_P in a casebase such that $C \cap AF_P = C_P \cap AF_P$. Then, there exists a supporting case to prove P for C or there exists a supporting case to disprove P for C.*

Proof: Let $C_{max} \in \mathcal{CB}$ be a case such that

- $C \cap AF_P = C_{max} \cap AF_P$, and
- there is no C' such that $C' \cap AF_P = C_{max} \cap AF_P$, and $C \cap C_{max} \subset C \cap C'$, and $\overline{C} \cap \overline{C_{max}} \subset \overline{C} \cap \overline{C'}$ where $\overline{C} = \mathcal{F} - C$.

Let IF_P be $((C \cap C_{max}) \cup (\overline{C} \cap \overline{C_{max}})) \cap PF_P$.
Then, $C \cap IF_P$
$= C \cap ((C \cap C_{max}) \cup (\overline{C} \cap \overline{C_{max}})) \cap PF_P$

$= ((C \cap C \cap C_{max}) \cup (C \cap \overline{C} \cap \overline{C_{max}})) \cap PF_P$
$= ((C \cap C_{max}) \cup \emptyset) \cap PF_P$
$= (C \cap C_{max}) \cap PF_P$
Similarly, $C_{max} \cap IF_P = (C \cap C_{max}) \cap PF_P$. Thus, $C \cap IF_P = C_{max} \cap IF_P$.

The second condition for C_{max} means that for every case $C' \in \mathcal{CB}$ except C_{max} which satisfies $C' \cap AF_P = C_{max} \cap AF_P$, $C \cap C_{max} \not\subset C \cap C'$ or $\overline{C} \cap \overline{C_{max}} \not\subset \overline{C} \cap \overline{C'}$.

Case 1: Suppose that $C \cap C_{max} \not\subset C \cap C'$.

Then, there exists a factor F such that $F \in C \cap C_{max}$ but $F \notin C \cap C'$.

This means that $F \in IF_P$. Moreover, $F \in C$ leads to $F \in C \cap IF_P$.

However, since $F \notin C \cap C'$ and $F \in C$, $F \notin C'$. This means $F \notin C' \cap IF_P$.

Therefore, $C \cap IF_P \neq C' \cap IF_P$.

Case 2: Suppose that $\overline{C} \cap \overline{C_{max}} \not\subset \overline{C} \cap \overline{C'}$.

Then, there exists a factor F such that $F \in \overline{C} \cap \overline{C_{max}}$ but $F \notin \overline{C} \cap \overline{C'}$.

This means that $F \in IF_P$. Moreover, $F \in \overline{C}$ (in other words, $F \notin C$) leads to $F \notin C \cap IF_P$.

However, since $F \notin \overline{C} \cap \overline{C'}$ (or equivalently $F \notin \overline{C}$ or $F \notin \overline{C'}$) and $F \in \overline{C}$, $F \notin \overline{C'}$ (thus $F \in C'$). This means that $F \in C' \cap IF_P$.

Therefore, $C \cap IF_P \neq C' \cap IF_P$.

In either case, for every case $C' \in \mathcal{CB}$ except C_{max} which satisfies $C' \cap AF_P = C_{max} \cap AF_P$, $C \cap IF_P \neq C' \cap IF_P$.

Suppose that $C_{max} \in P^+$. Then, for every case $C_{P^-} \in P^-$, either $C \cap AF_P \neq C_{P^-} \cap AF_P$, or $C \cap IF_P \neq C_{P^-} \cap IF_P$. Therefore, C_{max} is a supporting case to prove P for C.

On the other hand, suppose that $C_{max} \in P^-$. In a similar way, C_{max} is a supporting case to disprove P for C. □

Proof of Theorem 2:
Let C be the current case. For every basic proposition ψ, we take the following set of abducibles Θ_ψ and set of clauses S_ψ.

- Suppose that there is no case C_ψ such that $C \cap AF_\psi = C_\psi \cap AF_\psi$. In this case, Θ_ψ be $\{imp^*(F, \psi) | F \in \mathcal{F}\}$ and S_ψ be $\emptyset$.
- Suppose that there is a case C_ψ such that $C \cap AF_\psi = C_\psi \cap AF_\psi$. From Lemma 1, there is a supporting case and a set of possible important factors to prove ψ or disprove ψ for C. Especially, if $\psi = P$, there is a supporting case C_{P+} and a set of possible important factors IF_{P+} to prove P. In this case, let Θ_ψ be $\{imp^*(F, \psi) | F \in IF_\psi\}$ and S_ψ be
  $\{pf(F, \psi) \leftarrow | F \notin IF_\psi\}$
  $\cup \{npf(F, \psi) \leftarrow | F \notin IF_\psi\}$.

Let T' be a set of clauses obtained by

- removing all the clauses of the form:
  $pf(F, \psi) \leftarrow \sim imp^*(F, \psi)$
  and
  $npf(F, \psi) \leftarrow \sim imp^*(F, \psi)$, and
- adding $\bigcup_{\psi \in \mathcal{P}} S_\psi$.

Let M be $min(\Pi_{T'})$.

We first show that $M \models P(id(C_{P+}))$. Suppose that $M \not\models P(id(C_{P+}))$. There is only one clause for $P(id(C_{P+}))$ denoted as:
$P(c_{P+}) \leftarrow$
$af(A_1, P), ..., af(A_k, P)$,
$naf(A_{k+1}, P), ..., naf(A_l, P)$,
$pf(F_1, P), ..., pf(F_m, P)$,
$npf(F_{m+1}, P), ..., npf(F_n, P)$,
where $c_{P+} = id(C_{P+})$. Then, since $M \not\models P(c_{P+})$, at least, one of the following hold.

1. There exists $af(A_i, P)(1 \leq i \leq k)$ such that $M \not\models af(A_i, P)$.
2. There exists $naf(A_i, P)(k+1 \leq i \leq l)$ such that $M \not\models naf(A_i, P)$.
3. There exists $pf(F_i, P)(1 \leq i \leq m)$ such that $M \not\models pf(F_i, P)$.
4. There exists $npf(F_i, P)(m+1 \leq i \leq n)$ such that $M \not\models npf(F_i, P)$.

The first and second condition implies $C \cap AF_{P+} \neq C_{P+} \cap AF_{P+}$ and the third and fourth condition implies $C \cap IF_{P+} \neq C_{P+} \cap IF_{P+}$. This contradicts with the condition of C_{P+} and IF_{P+}.

Therefore, $M \models P(c_{P+})$. Similarly, we can show that $M \not\models \bot$.

Let $M(\Theta)$ be $M \cup \Theta$ where $\Theta = \bigcup_{\psi \in \mathcal{P}} \Theta_\psi$. Then, since $\Pi_{T(\Theta)}^{M(\Theta)} = \Pi_{T'(\Theta)}$,
$min(\Pi_{T(\Theta)}^{M(\Theta)}) = min(\Pi_{T'(\Theta)}) = M \cup \Theta = M(\Theta)$
Therefore, $M(\Theta)$ is a generalized stable model of T. □

Proof of Theorem 3
1 to 2
Let $M(\Theta)$ be a generalized stable model for T and C_1,...,C_n be cases such that

$$M(\Theta) \models P([id(C_1), ..., id(C_n)]).$$

Then, there must be a clause for $P([X_1, ..., X_n])$ of the form:

$$P([X_1, ..., X_n]) \leftarrow B_1(X_1), ..., B_n(X_n)$$

such that for each $B_i(1 \leq i \leq n)$, $M(\Theta) \models B_i(id(C_i))$.

According to Theorem 1, each C_i is a supporting case to prove a basic proposition B_i for the current case.

Therefore, there exists a supporting rule R to prove a derived proposition for the current case of the form:
$B_1, ..., B_n \Rightarrow P$.

2 to 1
Let R be a supporting rule to prove a derived proposition for the current case of the form:
$B_1, ..., B_n \Rightarrow P$
For each $B_i(1 \leq i \leq n)$, there is a supporting case C_i to prove B_i for the current case. In the proof of Theorem 2,

we showed that we can choose each set of abducibles to prove each proposition B_i independently from other sets of abducibles to prove/disprove other propositions. This means that we can construct a generalized $M(\Theta)$ for a translated program T such that $M(\Theta) \models B_i(id(C_i))$. Since there is a translation of R of the form:

$$P([X_1, ..., X_n]) \leftarrow B_1(X_1), ..., B_n(X_n),$$

and a body of the clause is satisfied,

$$M(\Theta) \models P([id(C_1), ..., id(C_n)]).$$

□

Proof of Theorem 4
1 to 2
Let $M(\Theta)$ be a generalized stable model for T and $C_1, ..., C_m$ be cases such that

$$M(\Theta) \models n\_P([id(C_1), ..., id(C_m)]).$$

Then, there must be a clause for $n\_P([X_1, ..., X_m])$ of the form:

$$n\_P([X_1, ..., X_m]) \leftarrow n\_B_1(X_1), ..., n\_B_m(X_m)$$

such that for each $B_i(1 \leq i \leq m)$, $M(\Theta) \models B_i(id(C_i))$.

Similar to the proof of Theorem 1, we can show that each C_i is a supporting case to disprove a basic proposition B_i for the current case. Then, for every rule $R_i(1 \leq i \leq m)$ in the rules of P, there exists a basic proposition B_i such that the current case is negative w.r.t. B_i. Therefore, the current case is negative w.r.t. P.

2 to 1
Let $B_i(1 \leq i \leq m)$ be a proposition in the condition of every rule of P, R_i, such that C_i is a supporting case to disprove B_i for the current case.

Similar to the proof of Theorem 2, we can show that we can choose each set of abducibles to disprove each proposition B_i independently from other sets of abducibles to prove/disprove other propositions. This means that we can construct a generalized $M(\Theta)$ for a translated program T such that $M(\Theta) \models n\_B_i(id(C_i))$. Since there is a translated clause to disprove P of the form:

$$n\_P([X_1, ..., X_m]) \leftarrow n\_B_1(X_1), ..., n\_B_m(X_m),$$

and the body of the clause is satisfied,

$$M(\Theta) \models n\_P([id(C_1), ..., id(C_m)]).$$

□