Transformation from PROLEG to a Bipolar Argumentation Framework

Tatsuki KAWASAKI [a] and Sosuke MORIGUCHI [a] and Kazuko TAKAHASHI [a,1]
[a] *Kwansei Gakuin University, Japan*

Abstract. We describe a transformation from the legally descriptive language PROLEG to a bipolar argumentation framework (BAF). PROLEG is a language consisting of general rules and exceptions and employs the negation-as-failure inference rule. In the transformation, each atom is transformed into an argument and the relationships of each atom are transformed into attack or support relationships in BAF. We formalize the transformation rules, and present the semantics of the transformed BAF so that the meaning of legal reasoning is preserved. We prove that the answer set of the PROLEG program coincides with the set of accepted arguments in BAF.

Keywords. bipolar argumentation framework, legal reasoning, PROLEG, normal logic programming, semantics

CEUR-WS.org/Vol-2171/paper_4.pdf

1. Introduction

Computational argumentation is useful when engaging in conflict resolution, aiding the attainment of agreement, and it finds many application fields [15]. One promising application is legal reasoning. In legal reasoning, computational support is very helpful both when making a judgment and also when analyzing or explaining the process involved [2].

Over the last few decades, legal reasoning using logic programming has been intensively investigated, but is burdensome for lawyers who lack familiarity with logic programming; they are unable to write legal knowledge appropriately or to understand the reasoning involved. One solution to this problem is the development of a descriptive language that encompasses the language of pure logic programming; lawyers readily understand descriptive language that represents legal knowledge. The PROLEG language was developed to this end [16]. PROLEG was originally developed to describe the Japanese "presupposed ultimate facts" theory (called Yoken-jijistu-ron in Japanese) of the Japanese civil code, and currently it is challenging to describe a penal code. PROLEG is an extension of Prolog, and features a negation-as-failure inference rule. PROLEG contains two types of sentences, general rules, and exceptions. The expressive power of PROLEG is the same as that of a normal logic program (NLP) with an answer set [17].

[1]Corresponding Author: Kwansei Gakuin University, 2-1 Gakuen, Sanda, Hyogo, 669-1337 Japan; E-mail:ktaka@kwansei.ac.jp

The following is a PROLEG program representing the penal code that defines the "crime of murder."[2] The first clause indicates the general rule and the second clause an exception. The text states that if the object is a human (not a dead body) and there exists both the action of murder and also the intention to murder, then the crime of murder has been committed unless a legitimate defense is available.

```
crime_of_murder <= human, action_of_murder, intention_to_murder.
exception(crime_of_murder,legitimate_defense).
```

If a case in which the atoms human, action_of_murder and intention_to_murder are proved to be true, and legitimate_defense is not proved to be true, then crime_of_murder is proved to be true, and is applied to the case. On the other hand, if legitimate_defense is proved to be true, then crime_of_murder is not proved to be true, and it is not applied to the case. This corresponds to the well-known negation-as-failure inference rule. A judge should explain the process of judgment to persuade those concerned with the justice. In such a legal situation, what is required is not only the outcome of judicial reasoning but also an explanation of the reasoning process or the cause-and-effect relationships of arguments used in reasoning. For example, people may wish to know why the crime_of_murder was not applicable; this may be because of a lack of evidence of intention_to_murder, or as the result of the fact that a legitimate_defense is proved to be true. Moreover, if it is possible to identify a reason, a lawyer should consider what s/he should do to make the crime_of_murder apply. For example, s/he will look for evidence of intention_to_murder if the lack causes to prevent the application of a crime_of_murder. However, this is not easy to understand by reference to the PROLEG program. Sentence delivered after judgment is regarded as a kind of argumentation. Since our final goal was to provide a tool that supports lawyers, we must create a representation allowing lawyers to easily grasp both the structure of an argument and the reason for judgment. Argumentation framework (AF) proposed by Dung [9] is a useful method to understand an argumentation process and reasons for a judgment. However, it does not explicitly show a specific argument that supports another argument. This motivated us to transform PROLEG into a bipolar argumentation framework (BAF) with a structure reflecting real arguments that consist of *pros* and *cons*. BAF is an extension of AF [1].

An AF is defined as a pair $\langle AR, ATT \rangle$ where AR is a set of arguments and ATT a binary relationship over AR, called *an attack*. Dung defined several semantics and showed that their relationships were the same as those of a normal logic program (NLP) [9]. He defined a transformation from NLP to AF, and showed that a stable model (an answer set) is equivalent to a stable extension of the transformed AF. He also considered the relationships between an NLP and an AF in terms of other semantics. In the transformation, each clause is transformed into a single argument, which consists of *a claim* and *a support*: for each clause, the head goal is transformed into a claim and the body goals into supports.

An NLP is a finite set of clauses in the following form:

[2]Note that the examples shown here are simplified versions of an actual penal code; the conditions per se are simplified and the legal terminology is not precise.

$$L :\text{-}\ L_1, \ldots, L_m, \mathit{not}\ L_{m+1}, \ldots, \mathit{not}\ L_{m+n}.$$

where $L, L_1, \ldots, L_{m+n}$ are atoms and *not* is a negation-as-failure operator [10]. For each atom h, the complement of h is denoted by h^*. Then, Dung showed that a logic program P was transformed into the AF $AF(P) = \langle AR, ATT \rangle$ as follows:

$$\begin{aligned} AR = \quad & \{(K, k) \mid K \text{ is a support for } k \text{ with respect to } P\} \\ & \cup \{(\{\neg k\}, \neg k) | k \text{ is a ground atom}\}. \end{aligned}$$

$$ATT = \{((K, H), (K', h')) \mid h^* \in K'\}.$$

In this transformation, for each clause, the body goals are unified into a single argument, and the goal h^* is not explicitly indicated in the transformed AF. Therefore, we cannot identify a specific goal that is attacked.

A BAF is an extension of an AF in which two types of relationships (attack and support) are defined over a set of arguments [1]. We can transform a PROLEG program into a BAF in a similar manner, but this does not suit our purpose that the causes and effects of individual arguments are clearly represented.

Here, we use a transformation that differs from that of Dung. We create an argument not for each clause but, rather, for each goal, and define attack or support relationships between the arguments.

In this paper, we develop a transformation from PROLEG to BAF and show that it is correct. More specifically, we give a semantics for the BAF obtained as a result of the transformation, and prove that a PROLEG answer set coincides with the set of accepted BAF arguments.

This paper is organized as follows. After explaining PROLEG in Section 2, we describe the BAF that we treat here and give its semantics in Section 3. Then, we develop the transformation rule from PROLEG to BAF in Section 4, and prove its correctness in Section 5. We compare our approach with those of others in Section 6. Finally, we offer concluding remarks and mention our future work in Section 7.

2. Legal Description Language: PROLEG

The PROLEG program P is defined as a pair $\langle \mathcal{R}, \mathcal{E} \rangle$, a finite set of rules, and a finite set of exceptions. Each rule is a Horn clause of the form $H \Leftarrow B_1, \ldots B_n$, where $H, B_1, \ldots, B_n$ are atoms (n may be 0, and we call such a rule *a fact rule* or simply *a fact*). Each exception is in the form exception(H, E).

For each rule R or exception E, we use the functions *head* and *body* s.t. $\mathit{head}(R) = H$ and $\mathit{body}(R) = \{B_1, \ldots, B_n\}$ if $R = H \Leftarrow B_1, \ldots, B_n$; $\mathit{head}(E) = H$ and $\mathit{body}(E) = \{B\}$ if $E = \text{exception}(H, B)$.

The semantics of the PROLEG program P are defined as an answer set (a set of ground atoms). M is the *answer set* of P iff M is the minimum model of P^M where P^M is the set of Horn clauses defined as $\{H \Leftarrow B_1, \ldots B_n \in \mathcal{R} \mid \forall E \in \mathcal{E},\ \text{if } \mathit{head}(E) = H \text{ then } \mathit{body}(E) \not\subseteq M\}$.

First, we define the relevant atoms for each atom. *The relevant atoms* of A are defined as $r_P(A) = \bigcup_{R \in \mathcal{R} \wedge \mathit{head}(R) = A} \mathit{body}(R) \cup \bigcup_{E \in \mathcal{E} \wedge \mathit{head}(E) = A} \mathit{body}(E)$.

The level function l_P for program P, from a atom to an integer, is defined as follows:

$$l_P(A) = \begin{cases} 1 & (r_P(A) = \emptyset) \\ \max_{B \in r_P(A)} l_P(B) + 1 & (\text{otherwise}) \end{cases}$$

When l_P is uniquely defined, P is said to be *acyclic*. Here, we focus on only acyclic PROLEG programs. Since l_P is unique, we state that atom A has a level n when $l_P(A) = n$.

3. Bipolar Argumentation Framework

Here, we deal with acyclic BAFs and give the semantics.

BAF is an extension of AF in which the two relationships of attack and support are defined over a set of arguments [1]. Different from usual BAF based on this definition, we define a support relationship between a power set of arguments and a set of arguments.

Definition 1. *BAF baf is defined as a triple* $\langle AR, ATT, SUP \rangle$ *where* AR *is a finite set of arguments,* $ATT \subseteq AR \times AR$ *and* $SUP \subseteq (2^{AR} \backslash \emptyset) \times AR$. *We define* $\mathrm{att}(B, A)$ *if* $(B, A) \in ATT$, *and* $\mathrm{sup}(\mathbf{A}, A)$ *if* $(\mathbf{A}, A) \in SUP$

Definition 2. The relevant set of arguments A *is defined as* $rel_A = \{B \mid \mathrm{att}(B, A)\} \cup \{B \mid B \in \mathbf{A} \wedge \mathrm{sup}(\mathbf{A}, A)\}$.

Definition 3. *For* $baf = \langle AR, ATT, SUP \rangle$, *the* height function h *of baf, from* AR *to a non-negative number, is defined as follows:*

- $h(A) = 0$ *if* $rel_A = \emptyset$.
- $h(A) = \max_{B \in rel_A}(h(B)) + 1$ *otherwise.*

Definition 4. *We say that baf is* acyclic *when the height function is definable.*

Figure 1 is a graphic representation of a BAF $\langle \{a, b, c, d, e\}\{(b, a), (e, d)\}\{(\{c, d\}, a)\}\rangle$. In this BAF, $rel_a = \{b, c, d\}$ and $h(b) = h(c) = h(e) = 0, h(d) = 1, h(a) = 2$.

Figure 1. Example of BAF.

We give a semantics for BAF based on labeling [5]. Usually, labeling is a function from a set of arguments to $\{in, out, undec\}$, but *undec* is unnecessary

here, because the BAF is acyclic. An argument labeled with *in* is considered to be an accepted argument. Labeling of a set of arguments is defined as follows.

Definition 5. *For baf* $= \langle AR, ATT, SUP \rangle$, labeling $\mathcal{L}$ *is a function from* AR *to* $\{in, out\}$. $\mathcal{L}(\mathbf{A}) = in$ *if* $\forall A \in \mathbf{A}, \mathcal{L}(A) = in$ *for all* $A \in \mathbf{A}$*;* $\mathcal{L}(\mathbf{A}) = out$*, otherwise.*

Complete labeling is determined so that legal judgment can be reflected. We put the label *in* to the argument that is not attacked nor supported by any argument. When an argument is attacked and also supported, an attack is considered to be stronger than a support. We put the label *out* to the argument that is attacked by an argument with a label *out* and at the same time supported by an argument with a label *out*.

Definition 6. *For baf* $= \langle AR, ATT, SUP \rangle$*, labeling* $\mathcal{L}$ *is* complete *iff the following conditions are satisfied for any argument* $A \in AR$*.*

- $\mathcal{L}(A) = in$ *if* $(\forall B \in AR, \neg \mathrm{att}(B, A)) \wedge (\forall \mathbf{A} \subseteq AR, \neg \mathrm{sup}(\mathbf{A}, A))$.
- $\mathcal{L}(A) = in$ *if* $(\forall B \in AR, \mathrm{att}(B, A) \Rightarrow \mathcal{L}(B) = out) \wedge (\exists \mathbf{A} \subseteq AR, \mathrm{sup}(\mathbf{A}, A) \wedge \mathcal{L}(\mathbf{A}) = in)$.
- $\mathcal{L}(A) = out$*, otherwise.*

Figure 2 illustrates examples of the complete labelings to four BAFs, respectively. In the figure, the straight arrow indicates an attack relation and the wavy arrow indicates a support relation.

Figure 2. Examples of BAFs with complete labelings.

Theorem 1. *For any acyclic BAF, there is one and only one complete labeling.*

Proof. We now prove the uniqueness of this labeling. Assume that two complete labelings $\mathcal{L}_1$ and $\mathcal{L}_2$ exist. The proof is by induction on the height n of arguments.

If $n = 0$, the relevant set of A is an empty set. This means that A is neither attacked nor supported, thus $\mathcal{L}_1(A) = \mathcal{L}_2(A) = in$ from the definition of complete labeling.

When $n > 0$, we proceed by induction; assume that for any argument A' of height less than n, $\mathcal{L}_1(A') = \mathcal{L}_2(A')$ holds. Then, we prove that for any argument A of height n, $\mathcal{L}_1(A) = \mathcal{L}_2(A)$ holds. From the definition of the height of an argument, there should be an argument B that fulfills $\mathrm{att}(B, A)$, or a non-empty set of arguments $\mathbf{A}$ that fulfills $\mathrm{sup}(\mathbf{A}, A)$. Since the heights of B and $\mathbf{A}$ are both less than n, $\mathcal{L}_1(B) = \mathcal{L}_2(B)$ and $\mathcal{L}_1(A') = \mathcal{L}_2(A')$ hold for each $A' \in \mathbf{A}$ (from the induction hypothesis). If $\mathcal{L}_1(B) = out$ and $\mathcal{L}_1(A') = in$ for some $\mathbf{A}$, then $A' \in \mathbf{A}$, and $\mathcal{L}_1(A) = in$; and, in the same manner, $\mathcal{L}_2(A) = in$. Otherwise, $\mathcal{L}_1(A) = out$ and $\mathcal{L}_2(A) = out$. Therefore, $\mathcal{L}_1(A) = \mathcal{L}_2(A)$.

Hence, for any argument A, $\mathcal{L}_1(A) = \mathcal{L}_2(A)$; thus, $\mathcal{L}_1 = \mathcal{L}_2$. Since we can thus perform complete labeling, such labeling must be unique. □

4. Transformation

4.1. Transformation rule

We here transform a PROLEG program into a BAF. Atoms, rules, and exceptions in P are transformed into arguments, supports, and attacks, respectively.

We add two types of arguments to the BAF that do not occur as explicit atoms in PROLEG. One is an argument reflecting the *absence* of any inference rules in PROLEG. In PROLEG, an atom H that does not appear in the head of any rule or exception is not in the answer set. On the other hand, arguments that are neither supported nor attacked are labeled *in*. To fill the gap, we add the argument absence(H) that attacks H. We call this argument *an absence argument.*

We also add arguments showing the *existence* of fact rules. For a fact rule (i.e., a rule in the form $H \Leftarrow$), there is no arguments that support H in BAF; whereas any support is a binary relationship. Therefore, we add an argument existence(H) that supports H. We call this argument an *existence argument.*

Definition 7. *Transformation from the PROLEG program* $\langle \mathcal{R}, \mathcal{E} \rangle$ *to BAF* (AR, ATT, SUP) *is defined as follows.*

- $Atom = \bigcup_{R \in \mathcal{R}} (head(R) \cup body(R)) \cup \bigcup_{E \in \mathcal{E}} (head(E) \cup body(E))$
- $Rule = \{(body(R), head(R)) \mid R \in \mathcal{R} \wedge body(R) \neq \emptyset\}$
- $Exc = \{(B, H) \mid \text{exception}(H, B) \in \mathcal{E}\}$
- $Existence = \{H \mid H \Leftarrow \in \mathcal{R}\}$
- $ExistenceSupport = \{(\{\text{existence}(H)\}, H) \mid H \in Existence\}$
- $Absence = Atom \backslash (\{head(R) \mid R \in \mathcal{R}\} \cup \{head(E) \mid E \in \mathcal{E}\})$
- $AbsenceAttack = \{(\text{absence}(B), B) \mid B \in Absence\}$
- $AR = Atom \cup \{\text{existence}(H) \mid H \in Existence\} \cup \{\text{absence}(B) \mid B \in Absence\}$
- $ATT = Exc \cup AbsenceAttack$
- $SUP = Rule \cup ExistenceSupport$

Clearly, we can process the transformation with a linear order of set operations of the size of P.

4.2. Examples

Example 1.

Consider the following PROLEG program (Prog1):

```
p <= q1,q2.
exception(q1,r).
q2<=.
r<=.
```

It is transformed into the following BAF:

$\langle \quad \{p, q_1, q_2, r, ex(q_2), ex(r)\},\ \{(r, q_1)\},$
$\{(\{q_1, q_2\}, p), (\{ex(q_2)\}, q_2), (\{ex(r)\}, r)\} \quad \rangle.$

Arguments q_1 and q_2 together support the argument p. The existence arguments $ex(q_2)$ and $ex(r)$ are created to support q_2 and r, respectively. Figure 3 shows a graphical representation of the BAF, with the complete labeling. Since $\mathcal{L}(q_1) = out$ and $\mathcal{L}(q_2) = in$, the label of the combined argument $\mathcal{L}(\{q_1, q_2\}) = out$. Since p is supported by $\{q_1, q_2\}$, $\mathcal{L}(p) = out$.

Figure 3. BAF for Prog1.

Example 2.

We now discuss a PROLEG program dealing with murder case.

The first part describes the penal code relevant to murder. The first set of rules and exceptions state that if the object is a human (not a dead body) and there exists both the action of murder and also the intention to murder, then the crime of murder has been committed unless a legitimate defense is available. The second set of rules and exceptions state that if the accused is infringed, and takes emergency, necessary, and appropriate action to defend himself/herself, then this is a legitimate defense, unless there is no aggressive intention to harm the deceased. The remaining part deals with the facts in evidence.

```
% rules regarding crime_of_murder
crime_of_murder <= human, act_of_murder, intention_to_murder.
exception(crime_of_murder,legitimate_defense).

legitimate_defense <=
  infringement, emergency, necessity, appropriateness,
  defense_intention.
exception(legitimate_defense,aggressive_intention_to_harm).

% facts
human <=.
act_of_murder <=.
intention_to_murder  <=.
infringement  <=.
emergency <=.
necessity <=.
appropriateness <=.
```

```
defense_intention <=.
```

For each proposition, a general rule defining that proposition together with the exceptions are transformed into the BAF using the transformation rule. If there exists a fact, then a corresponding existence argument supporting the proposition is created. If an atom does not appear in the head of any rule or exception, then a corresponding absence argument attacking the proposition is created in the transformed BAF.

We show the graphical representation of the transformed BAF in Figure 4.

Figure 4. Graphical representation for a transformed BAF of the murder case.

In this case, the argumentation process is explained as follows. Since the label of the absence argument ab(aggressive_intention_to_harm) is *in*, that of the aggressive argument aggressive_intention_to_harm is *out* (there is no aggressive intention to harm). Therefore, the label of the argument legitimate_defense is *in* (it is a legitimate defense). The argument crime_of_murder has one attacking argument, the label of which is *in*, and one supporting set of arguments, the label of which is *in*. Hence, the label of the argument crime_of_murder is *out* (the crime of murder is not applied).

5. Correctness of Transformation

Lemma 1. *For PROLEG program P and its transformation baf, each atom H in P retains its level in terms of height, that is, $l(H) = m$ in P iff $h(H) = m$ in baf.*

Proof. The proof proceeds by induction on the level m of atom H. Note that the levels of atoms commence at 1 whereas the heights of arguments commence at 0, and that every argument of which height is 0 is an existence argument or an absence argument. □

Theorem 2. *For PROLEG program P, let M be an answer set of P. Assume that $\mathcal{L}$ is the complete labeling of transformed BAF baf from P. Then, for each atom H in P, $H \in M$ iff $\mathcal{L}(H) = in$.*

Proof. The proof proceeds by induction on the level m of atom H.

When $m = 1$, H is either derived only by fact rules or no rules. If H is derived by fact rules, $H \in M$. Since these rules are transformed into supports with existence arguments (and no attacks to H exist), $\mathcal{L}(H) = in$. If H is not derived by any rules, $H \notin M$. In this case, H is not supported by any arguments but attacked by its absence argument; thus $\mathcal{L}(H) = out$. Therefore, $H \in M$ iff $\mathcal{L}(H) = in$.

For $m \geq 2$, assume that for any $k < m$, the following proposition holds: for any atom H' of level k, $H' \in M$ iff $\mathcal{L}(H') = in$. Since the level of H is greater than 1, H is derived by rules and/or exceptions of P.

First, we prove that if $H \in M$ then $\mathcal{L}(H) = in$. Since M is the minimum model, there exists a rule $H \Leftarrow B_1, \ldots, B_n$ such that, for each i, $B_i \in M$, and there are no exceptions exception(H, B) such that $B \in M$. In *baf*, the rule is transformed into sup$(\{B_1, \ldots, B_n\}, H)$. The levels of B and $B_1, \ldots, B_n$ are less than m, and from the induction hypothesis, for each i, $\mathcal{L}(B_i) = in$. Also, for any exceptions exception(H, B), the level of B is less than m, and $\mathcal{L}(B) \neq in$ since $B \notin M$ (from the cotraposition of the hypothesis). Therefore, $\mathcal{L}(H) = in$.

Next, we prove that if $\mathcal{L}(H) = in$, then $H \in M$. Since H is derived by some rule and/or exception, H is supported and/or attacked by some argument. This means that if $\mathcal{L}(H) = in$, then, for any argument B satisfying att(B, H), $\mathcal{L}(B) \neq in$. These attacks are generated from the exceptions exception(H, B), and from the induction hypothesis, $B \notin M$. Also, there is a support sup$(\{B_1, \ldots, B_n\}, H)$ such that $\mathcal{L}(B_i) = in$ for each i. Such a support is generated from the rule $H \Leftarrow B_1, \ldots, B_n$. Since the levels of $B_1, \ldots, B_n$ are less than m, $B_i \in M$ from the induction hypothesis. Since M is the minimum model (and no exceptions are satisfied), $H \in M$.

Thus, for any level m, $H \in M$ iff $\mathcal{L}(H) = in$. □

6. Discussion

Although transformation from NLP to AF is well known, no works can be found on that from NLP to BAF, to the best of authors' knowledge. The significant issue on the transformation is to give a semantics to the obtained BAF so that the meaning of legal reasoning is preserved. Here, we discuss existing works on the semantics of BAF.

Cayrol et al. investigated the semantics of BAF, defining several types of indirect attacks by combining an attack with supports. They also defined several types of extensions [6,7]. Then, they introduced the concept of "coalition" (a set of arguments) and defined a meta-AF using such a coalition [7]. The idea was to reduce a BAF to an AF by deleting the support relationships between arguments in the same coalition. Since this meta-AF can be regarded as a normal AF, a semantics can be defined. An argument in BAF is accepted if it is included in an accepted coalition of the meta-AF.

This method is problematic when it is applied to the BAF obtained by our transformation from PROLEG. Consider the following PROLEG program:

[PROLEG program: P_{PROLEG}]

```
p <= q1.
p <= q2.
exception(q1,r).
q2<=.
r<=.
```

The BAF obtained by our transformation is as follows:
$\langle \quad \{p, q_1, q_2, r, ex(q_1), ex(r)\},\ \{(r, q_1)\},$
$\{(\{q_1, q_2\}, p), (\{ex(q_2)\}, q_2), (\{ex(r)\}, r)\} \quad \rangle,$
which is shown in Figure 5. We obtain $\mathcal{L}(p) = \mathcal{L}(q_2) = \mathcal{L}(r) = in, \mathcal{L}(q_1) = out$. Therefore, p is accepted.

Figure 5. BAF for P_{PROLEG} obtained by our transformation.

Assume that this BAF is given. We construct coalition AF based on the definition in [7]. We now have two coalitions C_1 and C_2. C_1 is a coalition of r and $ex(r)$ and C_2 is a coalition of p, q_1, q_2 and $ex(q_2)$. Then, we obtain the meta-AF, $AF = \langle \{C_1, C_2\}, \{(C_1, C_2)\} \rangle$. Since $\mathcal{L}(C_1) = in, \mathcal{L}(C_2) = out$ in this meta-AF but C_2 is not accepted. Therefore, p is not accepted (Figure 6).

Figure 6. Coalition AF based on Cayrol's method.

According to [17], an NLP translated from P_{PROLEG} is as follows:

[Translated NLP from P_{PROLEG}: P_{NLP}]

```
p <= q1.
p <= q2.
q1 <= not r.
q2<=.
r<=.
```

Since the answer set of this program is $\{p, q_2, r\}$, p should be accepted in the corresponding BAF.

Using the semantics of coalition framework, all arguments in the coalition C_2 are considered to be attacked. A coalition is formed by unifying arguments, without considering the meaning of the original PROLEG program. Thus, we may have an accepted argument that is counterintuitive. On the other hand, in our method, p is accepted (Figure 5). Therefore, our method is more appropriate on dealing with legal knowledge.

In a coalition framework, an argument corresponds to a set of goals. Other frameworks in which support relation is handled have similar structures [11,13]. In these works, each support relationship is embedded in a single argument and only an attack relationship is defined between arguments. It is preferable, in a legal scenario, that the causes and effects of individual arguments are clear.

Oren and Norman developed an evidential argumentation by introducing a special argument, corresponding to an environment, into BAF [12]. This concept is similar to the existence argument of our method. The difference is that the cited authors introduced a single argument from which both attack and support relationships arise. In contrast, we create an existence or an absence argument for each fact. The structure of the evidential argumentation is more compact than ours, but the individual evidence for each fact is unclear.

Brewka et al. proposed an Abstract Dialectical Framework (ADF) as a generalization of the AF of Dung [3,4]. In the ADF, each node is associated with an acceptance condition depending on the parent nodes, and each link has an individual strength. The NLP can be represented as an associated ADF of which the nodes are atoms of the NLP. A bipolar ADF is a subclass of ADF in which each link is either attacked or supported depending on the polarity of its strength. This represents the BAF, and the BAF transformed from PROLEG may be considered as an instantiation of an ADF. It is interesting to investigate whether a semantics of ADF can be straightforwardly applied to the BAF transformed from PROLEG.

7. Conclusion

We developed a transformation from PROLEG to BAF and proved that it was correct. The resulting BAF reflects a structure of the reasoning and causality among ground atoms. We gave a semantics to the BAF that allows legal reasoning,

and proved that the answer set of PROLEG coincides with the set of accepted arguments in the BAF. We implemented the transformation using Prolog.

Here, we handled a stratified program, because well-defined Japanese civil and penal codes should be written in a stratified manner. Thus, the BAF obtained via transformation is acyclic and the labeling is unique. In future, we will transform a non-stratified NLP to a BAF, and discuss on other semantics. Then, we will compare these semantics with other formalizations including the coalition framework and ADF.

Acknowledgment

This work was supported by JSPS KAKENHI Grant Number JP17H06103.

References

[1] Amgoud, L., Cayrol, C., Lagasquie-Schiex, M. and Livet, P.: On bipolarity in argumentation frameworks. *International Journal of Intelligent Systems*, Volume 23,1062-1093 (2008).
[2] Bench-Capon, T., Prakken H. and Sartor, G.: Argumentation in Legal Reasoning. *Argumentation in Artificial Intelligence*, 363-382 (2009).
[3] Brewka, G. and Waltran, S.: Abstract Dialectical Frameworks. In *Proc. of KR2010*,102-111 (2010).
[4] Brewka, G. and Waltran, S. et al.: Abstract Dialectical Frameworks Revisited. *In Proc of IJCAI2013*, 803-809 (2013).
[5] Caminada, M.: On the Issue of Reinstatement in Argumentation. In *Proc. of JELIA2006*, 111-123 (2006).
[6] Cayrol, C. and Lagasquie-Schiex, M.: On the acceptability of arguments in bipolar argumentation frameworks. In *Proc. of ECSQARU2005*, 378-389 (2005).
[7] Cayrol, C. and Lagasquie-Schiex, M.: Coalitions of arguments in bipolar argumentation frameworks. In *Proc. of CMNA2007*, 14-20 (2007).
[8] Cayrol, C. and Lagasquie-Schiex, M.: Bipolarity in argumentation graphs: Towards a better understanding. *International Journal of Approximate Reasoning*, Volume 54, 876-899 (2013).
[9] Dung, P. M.: On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence*, Volume 77, 321-357 (1995).
[10] Gelfond, M. and Lifschitz, V.: The stable model semantics for logic programming. In *Proc. of ICLP*, 1070-1080 (1988).
[11] Kowalski, R., Dung, P. M. and Toni, F.: Assumption-based argumentation framework. *Argumentation in Artificial Intelligence*, 199-218 (2009).
[12] Oren, N., and Norman, T. J.: Semantics for Evidence-Based Argumentation. In *Proc. of COMMA2008*, 276-284 (2008).
[13] Prakken, H.: An abstract framework for argumentation with structured arguments. *Argument & Computation*, 93-124 (2010).
[14] Prakken, H., Reed, C. and Walton, D.: Dialogues about the burden of proof. In *Proc. of ICAIL2005*, 115-124 (2005).
[15] Rahwan,I. and Simari,G.(eds.): *Argumentation in Artificial Intelligence*, Springer (2009).
[16] Satoh, K. et al.: PROLEG: the Presupposed Ultimate Fact Theory by PROLOG Technology. *Information Network Law Review*, Volume 10, 54-89 (2011).
[17] Satoh, K. et al.: On Generality of PROLEG Knowledge Representation. In *Proc. of JURISIN2012*, 115-128 (2012).