Attacking Legal Argument by Examining Stability of Case Citation with Goal-Dependent Abstraction

Yoshiaki OKUBO and Makoto HARAGUCHI
Division of Electronics and Information Engineering
Hokkaido University
N-13 W-8, Sapporo 060, JAPAN
TEL: +81-11-706-7261
FAX: +81-11-706-7808
E-mail : {yoshiaki, makoto}@db.huee.hokudai.ac.jp

Abstract

This paper proposes a computational method of attacking a case-based legal argument in a Hypo[1]-like system. For this purpose, we introduce a notion of *stability* of case citation. Our basic idea of stability is based on a reasonable requirement: if an old case is cited for a new case, the old case should be cited for each case similar to the new case as well. If a case citation satisfies the requirement, then the citation is said to be *stable*. We propose in this paper a computational method for attacking a legal argument from the viewpoint of stability of case citation. In order to examine whether a case citation is stable or not, we need to obtain cases similar to the new case. These are obtained with the help of *Goal-Dependent Abstraction* (GDA [5]). GDA is an algorithm for finding appropriate similarity between sort concepts according to our viewpoint. Based on the similarity found by GDA, we *hypothetically* create a set of cases similar to the new case. Then, each hypothetical case is examined whether we can cite the same old case for the hypothetical case. If we found that the old case cannot be cited for a hypothetical case, we attack the argument by pointing out unstableness of its case citation.

1 Introduction

Case-Based Reasoning (CBR) is well known as a very powerful reasoning mechanism in legal domain. One of important tasks in CBR is to retrieve an appropriate case for a given new case(problem) from a case-base. Inappropriate retrieval will make the reasoning result *unreliable*. Especially, this point becomes serious in a *case-based legal argumentation system* such as Hypo[1].

Given a new case to be decided, such a system tries to construct an argument A_s for a side s (plaintiff or defendant) by 1)citing an old case that shares some legal factors with the new case and 2)applying the conclusion of the cited case to the new one in the expectation that similar cases would have the same conclusion. For the argument A_s, an opponent side s' tries to attack it by pointing out some weakness in A_s and constructs a counter argument $A_{s'}$ for the side. Therefore, if an old case is inappropriately cited for the side s, the case citation can be viewed as a weakness in the argument A_s and will become a subject to be attacked by the opponent side s'. Giving attention to this aspect, this paper tries to formalize such an attack of argument. That is, we propose a computational method of attacking a legal argument by *refuting its case citation.*

For this purpose, we introduce a notion of *stability* of case citation. Our basic idea of stability is based on the following reasonable requirement:

> Assume we have constructed an argument for a new case c_{new} by citing an old case c_{old}. The old case c_{old} should be cited, whenever we try to construct an argument for a case that is *similar* to c_{new}.

Let us assume that for a case c'_{new} similar to c_{new}, another old case c'_{old} is cited. It is implied that although c_{new} and c'_{new} are similar, we might have different conclusions for them. In the worst case, these conclusions might contradict each other. Therefore, arguments based on such case citations are considered unreliable.

A coherent case citation satisfying the above requirement is said to be *stable*. In this paper, we try to attack an argument from the viewpoint of stability of its case citation. Especially, we investigate this kind of attack in a Hypo ??-like case-based legal argumentation system.

In such a system, if an old case c_{old} shares some legal factors with a given new case c_{new}, c_{old} is considered to be *on-point* to c_{new} and becomes a candidate to be cited for c_{new}. Intuitively speaking, this kind of case citation considers the shared legal factors to be important in order for a side to win. Among the candidates, a case that maximally

ICAIL-97, Melbourne, Australia © 1997 ACM 0-89791-924-6/97/06.$3.50

shares legal factors [1] is actually cited as a *most-on-point* one to c_{new}. In order to attack an argument based on this kind of case citation, we try to examine its stability. If the case citation is not stable, we attack the argument by pointing out its unstableness.

Let us assume that for a new case c_{new}, an argument for a side s has been constructed by citing an old case c_{old} as a most-on-point one to c_{new}, where a set of legal factors F_s is shared in both cases. For the examination of the stability, we need to obtain a set of cases that are *similar* to c_{new}. As mentioned above, in the case citation, F_s is considered to be important in order for the side s to win. In other word, the case citation is made by focusing on F_s. As many researchers have pointed out, similarity highly depends on our *viewpoint*, *purpose* or *context*. In order to obtain cases similar to c_{new}, therefore, we should take a similarity from the viewpoint of F_s into account. That is, we try to find similarity depending on the viewpoint of F_s that is important for s to win. We can find such a similarity with the help of *Goal-Dependent Abstraction* (GDA, for short) [5].

GDA is an algorithm for finding an appropriate similarity between sort concepts *depending on* a viewpoint (goal) on which we are focusing. That is, we find an appropriate similarity for the shared factors F_s by GDA. Based on the similarity, a set of cases similar to c_{new} is created *hypothetically*. For each hypothetical similar case, we examine whether the old case c_{old} can be cited as a most-on-point one. If we found a hypothetical similar case c'_{new} for which another old case c'_{old} favorable to the opponent side s' can be cited as a most-on-point one, the argument is attacked with a claim:

> Although c'_{new} is similar to c_{new} from the viewpoint of F_s that is important for the side s to win, c_{old} cannot be cited as a most-on point one to c'_{new}. Another old case c'_{old} having an opposite conclusion is most-on-point to c'_{new} and can construct a counter argument that is favorable to the opponent side s'. Therefore, the case citation focusing on F_s is not adequate for deciding c_{new}.

It should be emphasized that although this paper investigates a method for attacking an argument by examining the stability of its case citation, use of the stability is not limited to that purpose. The stability can be used as a criterion for case citation. That is, if we have several candidates of case citations, we would prefer a stable one to the others.

2 Preliminaries

We introduce in this section some terminologies.

We are assumed to have two kinds of predicates, *factual* and *legal* predicates. *Factual predicates* are used to represent a *raw* case. Therefore, they also contain general (commonly-used) concepts or relations that are *not specific* to legal domain. For example, we consider predicates such as `car_maker(toyota)` and `employed_by(tom, toyota)` to be factual ones. Given a set of factual predicates, we distinguish predicates denoting *sort concepts* from the others. In the above example, `car_maker(toyota)` would be distinguished from `employed_by(tom, toyota)`, since the former denotes a sort concept and the latter a relation. Predicate symbols denoting sort concepts are simply called *sort symbols*. For example, `car_maker` is a sort symbol.

On the other hand, *legal predicates* are used to represent a *legal* case. Therefore, they contain only concepts or relations that are *specific* to legal domain. Each legal predicate is assumed to be propositional and is called a *legal factor*. For example, predicates (propositions) such as `disclosure_in_negotiations` and `secret_disclosed_outsider` used in Hypo[1] are legal factors.

We have a *knowledge-base* in order to relate factual predicates with legal factors. A knowledge-base consists of Horn-rules that are of the form

$$H \leftarrow B_1 \wedge \cdots \wedge B_n \qquad (n \geq 1),$$

where H is a legal factor or a factual predicate and B_i is a factual predicate. We call H the *conclusion* and B_i a *premise* of the rule. For a rule R, its conclusion and premises are denoted by $conc(R)$ and $pre(R)$, respectively. Intuitively speaking, our knowledge-base can be viewed as a variation of *Dimension Knowledge* in Hypo[1].

As mentioned above, we deal with two types of cases, raw and legal cases. A *raw case* is represented as a set of ground factual predicates. Each raw case c has its corresponding legal case denoted by $legal(c)$. More formally speaking, the legal case is defined as

$$legal(c) = \{f \mid f \text{ is a legal factor such that } c \cup \mathcal{KB} \vdash f\},$$

where $\mathcal{KB}$ is a knowledge-base. In a word, the legal case is the set of legal factors that hold in the raw case.

3 Constructing Legal Arguments Based on Claim Lattice

In this section, we discuss how a legal argument can be constructed in a Hypo-like legal argumentation system.

Given a new case to be decided, such a system constructs a legal argument for a side (plaintiff or defendant) with the help of *claim lattice*. Based on the claim lattice, an old case is first retrieved as a *most-on-point* one to the new case. Then a legal argument is constructed by applying the conclusion of the old case to the new case.

3.1 Case-Base

A *case-base* is a collection of *precedents*. Each precedent is represented by a triple of a legal case, an identifier (name) of the case and a side (π:plaintiff or δ:defendant) that won in the case. For example, a precedent recording that a plaintiff π won in a legal case C whose identifier is I, is represented by (I, C, π). The identifier of C is often referred as $id(C)$.

[1] The underlying ordering is based on set-inclusion.

3.2 Claim Lattice

Given a new case to be decided, a *claim lattice* is constructed in order to make a decision for the new case. The claim lattice is used to select an old legal case (precedent) to be cited for the decision. We discuss here how we construct a claim lattice for a new case.

Let c_{new} be a new case to be decided, $\mathcal{CB}$ be a case-base and $\mathcal{KB}$ be a knowledge-base. In this paper, each new case is assumed to be a raw case. A claim lattice is a directed acyclic graph. In the lattice, we have a special node N_0 (root node) which contains the legal case of c_{new}, that is,

$$legal(c_{new}) = \{f \mid f \text{ is a legal factor such that } c_{new} \cup \mathcal{KB} \vdash f\}.$$

Any other node N_i contains a set of legal factors (denoted by $LF(N_i)$) and a set of identifiers of legal cases with their sides (denoted by $ID(N_i)$) in $\mathcal{CB}$. In a word, the node N_i can be viewed as the collection of old cases each of which *exactly* shares the legal factors $LF(N_i)$ with $legal(c_{new})$. More formally speaking, for each (non-root) node N_i, $LF(N_i)$ and $ID(N_i)$ satisfy the following conditions:

1. Each case identifier in $ID(N_i)$ can be found in $\mathcal{CB}$.
2. For any case identifiers $id(C_1)$ and $id(C_2)$ in $ID(N_i)$,

$$legal(c_{new}) \cap C_1 = legal(c_{new}) \cap C_2 = LF(N_i) \ (\neq \phi).$$

For any two node N_i and N_j, if $LF(N_i) \supset LF(N_j)$, then a directed edge from N_i to N_j is made. Thus, we can construct a claim lattice for a given new case.

3.3 On-Pointness of Cases

Based on a claim lattice for a new case, we can retrieve an old case that is *most-on-point* to the new case according to a side on which we are.

Let c_{new} be a new case to be decided and C be a legal case in a case-base. If C shares some legal factors with the legal case of c_{new} (that is, $legal(c_{new})$), then C is said to be *on-point* to c_{new}. More precisely speaking, the set of legal cases that are on-point to the new case, denoted as $OP_{c_{new}}$, are defined as follows:

$$OP_{c_{new}} = \{C \mid (id(C), C, s) \in \mathcal{CB} \text{ and } legal(c_{new}) \cap C \neq \phi\}.$$

where $\mathcal{CB}$ is a case-base and s is either π or δ. Intuitively speaking, $OP_{c_{new}}$ is the set of legal cases that are relevant to c_{new} in the sense that some legal factors are shared with $legal(c_{new})$. For two cases C_1 and C_2 in $OP_{c_{new}}$, if $(C_1 \cap legal(c_{new})) \subset (C_2 \cap legal(c_{new}))$, then C_2 is said to be *more-on-point* than C_1.

According to the on-pointness, the set of *most-on-point* cases, denoted as $MOP_{c_{new}}$, is defined as

$$MOP_{c_{new}} = \{C \mid C \in OP_{c_{new}} \text{ and there is no } C' \text{ in } OP_{c_{new}} \text{ that is more-on-point than } C\}.$$

For a new case c_{new}, the identifier of each case on-point to c_{new} is contained in a claim lattice for the new case and others are never. Moreover, the identifier of each most-on-point case can be found in nodes in the lattice that are immediately connected to the root node.

3.4 Legal Argument Based on Most-On-Point Case

With the help of claim lattice, we can construct a legal argument for a new case according to a side on which we are.

A claim lattice for a new case c_{new} is firstly constructed. Then an argument for c_{new} that is favorable to a side s is obtained by citing an old case C_{old} from $MOP_{c_{new}}$ in which the side s won and applying the conclusion of C_{old} to c_{new}. Let us assume that $legal(c_{new})$ and C_{old} exactly share a set of legal factors F_s. It is claimed in the argument that

> Since the new case c_{new} can be considered similar to the old case C_{old} in the sense that they share F_s, c_{new} would have the same conclusion as C_{old} has.

In the next section, we propose a method for attacking a legal argument obtained in such a manner.

4 Attacking Legal Argument by Examining Stability of Case Citation

Let us assume that a legal argument for a new case c_{new} has been obtained by citing an old case C_{old}. Constructions of legal arguments in case-based manner are carried out in the expectation that similar cases would have the same conclusion. Therefore, if we try to construct an argument for another new case c'_{new} that is similar to c_{new}, the argument for c'_{new} should be obtained by citing the same old case C_{old}. A citation of another old case implies that although c'_{new} and c_{new} are similar, we might have different conclusions for them. In the worst case, these conclusions might contradict each other. Therefore, the authors consider that *stability* (coherence) of case citation should be taken into account in order to obtain strong (reliable) arguments. In other words, an argument based on unstable case citation can be attacked by pointing out the unstableness of case citation. In this section, we present a computational method for attacking arguments from the viewpoint of stability of case citation.

4.1 Stability of Case Citation

We present here a formal definition of *stability* of case citation.

Definition 1 (Stability of Case Citation)
Let us assume that a side s has constructed a legal argument for a new case c_{new} by citing an old case C_{old} as a most-on-point one to c_{new}. The citation of C_{old} is said to be *stable* iff for each case c'_{new} similar to c_{new}, the side s can cite the old case C_{old} as a most-on-point one as well. ■

In order to attack a legal argument for a new case that an opponent constructed. we try to examine whether its case citation is stable or not. If we found that the citation is not stable. then it becomes a subject to be pointed out in attacking the argument. For the examination, we need to find a set of cases that are similar to the new case. Such similar cases can be obtained with the help of Goal-Dependent Abstraction.

4.2 Examining Stability of Case Citation with GDA

We discuss here how we examine stability of case citation. Before precise discussions, we present a framework of *Goal-Dependent Abstraction* (GDA, for short) that plays a very important role for the examination.

4.2.1 GDA : Goal-Dependent Abstraction

GDA [5] is an algorithm for finding *appropriate similarities* between sort concepts *according to a viewpoint.* Before giving a formal definition of appropriate similarity for a viewpoint, it would be helpful for the reader to provide its basic idea with a simple example.

Basic Idea:

Let us assume that we have the following legal rules:

"Male whose age is above twenty is permitted smoking."

"Female whose age is above twenty is permitted smoking."

"Female whose age is above sixteen is permitted marriage."

In the rules, "*male*" and "*female*" are sort concepts. From the rules, we know that a person whose age is above twenty can be permitted smoking irrespective of sex. In other words, we do not have to recognize any difference between sort concepts "*male*" and "*female*" when we consider "*permission of smoking*". In this sense. we can consider that "*male*" and "*female*" are similar each other. In other words, this similarity can be considered appropriate when we consider "*permission of smoking*".

On the other hand. when we consider "*permission of marriage*". we need to distinguish sort concepts "*male*" and "*female*". Female whose age is above sixteen can be permitted marriage. while male whose age is above sixteen cannot be so [2]. In this sense. we cannot consider that "*male*" and "*female*" are similar. although such a similarity seems to be appropriate when we consider "*permission of smoking*".

Thus our appropriateness of similarity highly depends on a subject (viewpoint) we are considering. Such a dynamic aspect of similarity is reflected in our framework of GDA. Given a subject we try to consider. our GDA algorithm computes an appropriate similarity for the subject. In the framework. a subject is represented in fact as a *goal to be proved.* Below we formally discuss the framework of GDA.

[2] Actually, male cannot be permitted marriage in Japan. unless his age is above eighteen.

Sort Mapping:

We first need to introduce a notion of *sort mapping* that represents a similarity between sort concepts.

Let S be a set of predicate symbols that denote sort concepts. Such predicate symbols are simply called *sort symbols.* A mapping $\varphi : S \mapsto S'$ is a *sort mapping*, where S' is a set of sort symbols not appearing in S. A sort mapping can easily be extended to a mapping between two first-order languages, $\varphi : L \mapsto L'$, where the sets of sort symbols in L and L' are S and S', respectively, and any non-sort symbol is mapped into itself under φ. Any expressions E_1 and E_2 are said to be *similar* under φ if $\varphi(E_1) = \varphi(E_2)$. Therefore, a sort mapping can be viewed as a representation of similarity between sort concepts. In what follows, the terms "sort mapping" and "similarity" are used to give the same meaning. Moreover, for an expression E', $\varphi^{-1}(E')$ is defined as $\{E \mid \varphi(E) = E'\}$. That is, $\varphi^{-1}(E')$ can be viewed as a class of similar expressions.

For example, let us consider Horn-rules

$$\texttt{permitted\_smoking(X)} \leftarrow \texttt{male(X)} \wedge \texttt{above\_twenty(X)} \quad \text{and}$$

$$\texttt{permitted\_smoking(X)} \leftarrow \texttt{female(X)} \wedge \texttt{above\_twenty(X)},$$

where `male` and `female` are sort symbols. If `male` and `female` are similar under a sort mapping φ, that is, $\varphi(\texttt{male}) = \varphi(\texttt{female}) = \alpha$, the two rules are mapped into the same rule

$$\texttt{permitted\_smoking(X)} \leftarrow \alpha\texttt{(X)} \wedge \texttt{above\_twenty(X)}$$

under φ. Furthermore, the rules are said to be similar under φ.

Abstraction Based on Sort Mapping:

The abstraction framework of Tenenberg [2] is used as the basis of our GDA. So we introduce here the abstraction framework. Exactly speaking, a modified version of Tenenberg's framework is presented.

Let $\varphi : L \mapsto L'$ be a sort mapping and T be a set of Horn-clauses in L. Assume that Rs is the set of rules and Fs is the set of facts in T. We can transform T into an abstract Horn-clause set according to the following definition.

Definition 2 (*SortAbs*)
$SortAbs_\varphi(T)$ defined as follows is called an abstract Horn-clause set of T based on φ:

$$SortAbs_\varphi(T) = AbsRules_\varphi(Rs) \cup AbsFacts_\varphi(Fs).$$

where

$$AbsRules_\varphi(Rs) = \{\varphi(R) \mid R \in Rs \text{ and for each } Bs \in \varphi^{-1}(\varphi(pre(R))). \text{ there exists } H \in \varphi^{-1}(\varphi(conc(R))) \text{ such that } Rs \vdash H \leftarrow Bs\} \quad \text{and}$$

$$AbsFacts_\varphi(Fs) = \{\varphi(F) \mid F \in Fs\}. \quad \blacksquare$$

Intuitively speaking. $\varphi(R)$ can be contained in $AbsRules_{\varphi}(Rs)$, if each rule similar to R under φ is provable from T.

Example 1
Let us assume we have the following set of Horn-clauses T:

```
T = { permitted_smoking(X) ←
                male(X) ∧ above_twenty(X),
      permitted_smoking(X) ←
                female(X) ∧ above_twenty(X),
      permitted_marriage(X) ←
                female(X) ∧ above_sixteen(X).
      male(john).
      above_twenty(john),
      female(mary).
      above_sixteen(mary) },
```

where male and female are sort symbols. Furthermore. let φ be a sort mapping such that φ(male) = φ(female) = α. Based on the mapping. we can obtain the following abstract Horn-clause set:

```
SortAbs_φ(T) = { permitted_smoking(X) ←
                          α(X) ∧ above_twenty(X),
                 α(john).
                 above_twenty(john).
                 α(mary),
                 above_sixteen(mary) }.
```

It should be noted that the rule R

```
permitted_marriage(X) ← female(X) ∧ above_sixteen(X)
```

has no corresponding abstract rule. The reason is that a rule similar to R. that is.

```
permitted_marriage(X) ← male(X) ∧ above_sixteen(X),
```

is not provable from the rule set of T. ■

In general. thus. given a sort mapping φ and a Horn-clause set T. a *part of* rules in T can be abstracted under φ *to construct the abstract rule set.* As mentioned above, for a rule R. if *all* rules similar to R under φ are provable from T. the rule can be abstracted. In other words. existence of an abstract rule R' guarantees existence of all similar rules that are mapped into R' under φ. Thus. rules that can be abstracted are dependent on a sort mapping (similarity) we provide beforehand. This observation has motivated us to propose GDA.

Appropriate Similarity between Sort Symbols:

Let T be a Horn-clause set. G be a provable goal (not sort predicates) from T and φ be a sort mapping. Furthermore assume T can be divided into the rule set Rs and the fact set Fs. From Definition 2. if $\varphi(G)$ is provable from $SortAbs_{\varphi}(T)$. we can observe the following fact:

> For each fact set Fs' such that $\varphi(Fs) = \varphi(Fs')$. the goal G is provable from $Rs \cup Fs'$.

For each of sorts s_1 and s_2, if the same property (or relation) holds, human beings often consider that s_1 and s_2 are similar from the viewpoint of the property. The above observation reflects such a natural aspect of similarity. That is, we would be able to find some similarity between sorts in Fs and Fs' from the viewpoint of G. Therefore, the authors consider that φ is an appropriate similarity for the goal G. Given a goal to be proved and a Horn-clause set, our GDA algorithm computes such an appropriate similarity (sort mapping) between sort symbols for the goal.

Our appropriateness of similarity for a given goal is formally defined as follows.

Definition 3 (Appropriate Similarity for Goal)
Let T be a set of Horn-clause, Rs be the rule set of T, G be a provable goal (not sort predicates) and $\mathcal{T}_{\mathcal{R}}(G)$ be a proof-tree of G from T. If the following condition holds, a sort mapping (similarity between sort symbols) φ is said to be appropriate for G:

$$\varphi(Rules(G)) \subseteq AbsRules_{\varphi}(Rs),$$

where $Rules(G)$ is the set of rules in Rs that are used in $\mathcal{T}_{\mathcal{R}}(G)$. ■

Example 2
Let us consider again the Horn-clause set T and the similarity φ in Example 1. The similarity φ is appropriate for a goal permitted_smoking(X), since each rule used to prove the goal can be abstracted based on φ. On the other hand, φ is not appropriate for another goal permitted_marriage(X), since the rule

```
permitted_marriage(X) ←
                female(X) ∧ above_sixteen(X)
```

necessary to prove the goal cannot be abstracted based on the similarity. It should be emphasized that this result matches the appropriateness that we have intuitively explained before. ■

4.2.2 Finding Similarity by GDA : Shared Legal Factors as Goal for GDA

Let c_{new} be a new case to be decided. Assume that by constructing a claim lattice for c_{new}. an old case C_{old} has been cited for a side s (opponent side) as a most-on-point one to c_{new}. where C_{old} shares the set of legal factors LF with the legal case $legal(c_{new})$. In the case citation. LF is considered to be important in order for s to win. That is. the opponent is focusing on LF in the citation. As many researchers have pointed out. similarities should be dependent on our viewpoint. purpose or context. Since we try to examine the stability of opponent's case citation by using cases *similar* to c_{new}. we should take a similarity into account from the viewpoint of LF on which the opponent focused in the citation. Therefore. we find an appropriate similarity for LF by GDA.

More precisely speaking. we compute a similarity as follows. Let $\mathcal{KB}$ be a knowledge-base. At first. a proof-tree of

LF from $\mathcal{KB} \cup c_{new}$ is constructed. Then, we identify the set of rules in $\mathcal{KB}$, denoted by $Rules(LF)$, that are used in the proof-tree. From Definition 3, an appropriate similarity φ for LF satisfies the following condition:

$$\varphi(Rules(LF)) \subseteq AbsRules_{\varphi}(\mathcal{KB}).$$

GDA computes such a similarity by a generate-and-test strategy. That is, candidates for similarities (sort mappings) are generated and then tested for their appropriateness according to the above condition. It should be noted that in order to prune useless candidate generations, GDA adopts a pruning method. More precise descriptions of GDA algorithm can be found in the literatures [3, 5].

4.2.3 Creating Hypothetical Similar Cases

Let us assume that we obtained an appropriate similarity φ by GDA according to the above procedure. Based on the similarity, we can *hypothetically* create a set of cases similar to the new case c_{new} in order to examine the stability of case citation.

The set of hypothetical similar cases $HypoSim_{\varphi}(c_{new})$ is formally defined as

$$HypoSim_{\varphi}(c_{new}) = \varphi^{-1}(\varphi(c_{new}))\backslash\{c_{new}\},$$

where "\" denotes the set-difference operator. It should be noted that each constant symbol in $c'_{new} \in HypoSim_{\varphi}(c_{new})$ denotes some *hypothetical* object.

As stated previously, for each hypothetical similar case c'_{new}, the legal factors LF for which φ was computed can be derived from $c'_{new} \cup \mathcal{KB}$. That is, each hypothetical case shares at least the legal factors with the old case cited in the opponent's case citation.

4.2.4 Examining Stability of Case Citation

Created the set of hypothetical similar case $HypoSim_{\varphi}(c_{new})$, we construct a claim lattice for each $c'_{new} \in HypoSim_{\varphi}(c_{new})$. Based on the claim lattice, then, we examine whether the old case C_{old} that the opponent cited for c_{new} can be cited as a most-on-point one to c'_{new} for the opponent's side. If C_{old} can be cited as a most-on-point one to each hypothetical case, the opponent's case citation can be considered stable. If not so, the citation is considered unstable and becomes a subject to be refuted in attacking the opponent's argument.

4.3 Attacking Legal Argument by Refuting Case Citation

Let us assume that for a new case c_{new}, a legal argument for an opponent side has been constructed by citing an old case C_{old} as a most-on-point one to c_{new}, where the set of shared legal factors is LF. According to the procedures explained previously, we can examine the stability of case citation. If we found the citation is unstable, we try to attack the opponent's argument by refuting the case citation.

Assume that for a hypothetical similar case c'_{new}, C_{old} cannot be cited as a most-on-point one. This means that there exists another old case C'_{old} that is more-on-point to c'_{new} than C_{old}. For such a C'_{old}, we have the following two possibilities:

- C'_{old} is favorable to the opponent side.
- C'_{old} is favorable to our side.

In the former, although the opponent's case citation is unstable, pointing out the unstableness would not be effective in attacking opponent's argument. Because an argument that is favorable to the opponent side can be constructed in the end. That is, unstableness in the former case does not seem so convincing.

On the other hand, in the latter, we can strongly attack the opponent's argument since although c'_{new} is similar to c_{new}, an argument favorable to *our side* can be constructed for c'_{new}. Such an argument for c'_{new} would be considered as a counter argument to the opponent's. It should be emphasized here that c'_{new} was created hypothetically based on the similarity that was obtained from the viewpoint of *the legal factors on which the opponent focused in his/her argument construction.* Therefore, we take unstableness only in the latter case into account to attack the opponent's argument.

In the examination of stability, if we find a hypothetical similar case c'_{new} for which we can construct an argument favorable to our side by citing an old case C'_{old} more-on-point than C_{old}, we attack the opponent's argument with a claim:

> Although c_{new} and c'_{new} are similar from the viewpoint of F_s that is important for the opponent to win, C_{old} cannot be cited as a most-on point one to c'_{new}. Another old case C'_{old} having an opposite conclusion is most-on-point to c'_{new} and can construct a counter argument that is favorable to our side. Therefore, the opponent's citation focusing on F_s is not adequate for deciding c_{new}.

4.4 Illustration of Argument Attack

Let us assume that we have a knowledge-base $\mathcal{KB}$ and a case-base $\mathcal{CB}$ shown in Figure 1 [3]. The knowledge base is represented as Prolog-rules. For example, a Prolog-rule of the form

$$\mathtt{q(X)} : -\mathtt{p_1(X)}, \ldots, \mathtt{pn(X)}$$

is interpreted as a Horn-rule of the form

$$\mathtt{q(X)} \leftarrow \mathtt{p_1(X)} \wedge \cdots \wedge \mathtt{pn(X)}.$$

In the figure, propositional predicates are legal factors and others are factual predicates, where we consider company, car_maker, car_dealer and bank to be sort symbols. That is, our GDA tries to find a similarity between these symbols. Furthermore, π and δ denote the sides of plaintiff and defendant, respectively.

Assume that we are given the following new (raw) case c_{new} to be decided:

[3] They were partly borrowed from [1].

$\mathcal{KB}$:

non_disclosure_agreement : –
company(X), side(X, π), company(Y), side(Y, δ),
made_a_secret(X, T), brought(T, X, Y),
entered_into_non_disclosure_agreement(Y, X, T).

secret_disclosed_outsider : –
company(X), side(X, π), outsider(Y, X),
made_a_secret(X, T), brought(T, X, Y).

security_measure : –
company(X), has_security_system(X).

competitive_advantage_gained : –
company(X), side(X, π), company(Y), side(Y, δ),
brought_to(T, X, Y), advantage_gained(Y, T).

disclosure_in_negotiations : –
company(X), side(X, π), company(Y), side(Y, δ),
made_a_secret(Z, X), had_negotiation(X, Y, Z),
brought(Z, X, Y).

vertical_knowledge : –
company(X), side(X, π), company(Y), side(X, δ),
made_a_secret(Z, X), vertical_information(Z),
brought(Z, X, Y).

vertical_information(Y) : –
bank(X), made_a_secrete(Y, X).

company(X) : –car_maker(X).

company(X) : –car_dealer(X).

company(X) : –bank(X).

$\mathcal{CB}$:

(*Automated_Systems*,
{disclosure_in_negotiations,
vertical_knowledge}, δ)

(*Space_Aero*,
{security_measure, competitive_advantage_gained,
disclosure_in_negotiations}, π)

(*Healy*, {secret_disclosed_outsider}, δ)

Figure 1: A knowledge-base and a case-base

c_{new} = { car_maker(tyt), side(tyt, π),
car_maker(nsn), side(nsn, δ),
car_maker(hnd), outsider(tyt, hnd),
made_a_secret(info, tyt),
brought(info, tyt, nsn),
brought(info, tyt, hnd),
had_negotiation(tyt, nsn, info),
non_disclosure_agreement(nsn, tyt, info) }

For the new case, its corresponding legal case $legal(c_{new})$ is

$legal(c_{new})$ = { disclosure_in_negotiations,
secret_disclosed_outsider,
non_disclosure_agreement }.

For the legal case, we can obtain the claim lattice shown in Figure 2. Based on the lattice, the plaintiff π cites *Space_Aero* case as a most-on-point one to the new case. We try to refute this case citation from the defendant side δ.

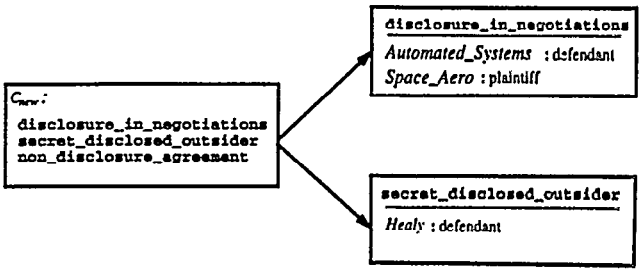

Figure 2: Claim lattice for a new case

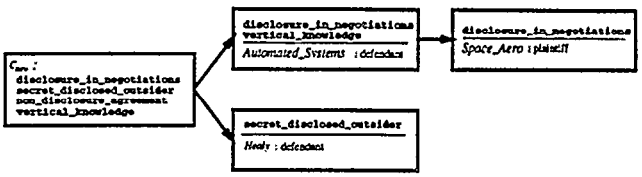

Figure 3: claim lattice for a hypothetical similar case

In the case citation, the legal factor

disclosure_in_negotiations

is considered to be important for π to win. Therefore, by considering the factor as a goal, GDA computes a similarity that is appropriate for the factor. In this case, a similarity φ such that

$$\varphi(\text{car\_maker}) = \varphi(\text{car\_dealer}) = \varphi(\text{bank})$$

can be obtained. Based on the similarity, let us consider the following hypothetical case c'_{new} similar to c_{new}.

c'_{new} = { bank(tyt), side(tyt, π),
car_dealer(nsn), side(nsn, δ),
car_dealer(hnd), outsider(tyt, hnd),
made_a_secret(info, tyt),
brought(info, tyt, nsn),
brought(info, tyt, hnd),
had_negotiation(tyt, nsn, info1),
non_disclosure_agreement(nsn, tyt, info) }

Figure 3 shows the claim lattice constructed for this hypothetical case. Based on the claim lattice, the defendant side can refute the plaintiff's case citation with the following claim:

> Although c_{new} and c'_{new} are similar from the viewpoint of the legal factor disclosure_in_negotiations that is important for the plaintiff to win, *Space_Aero* case cannot be cited as a most-on point one to c'_{new}. Another old case *Automated_Systems* having an opposite conclusion is most-on-point to c'_{new} and can construct a counter argument that is favorable to the defendant side. Therefore,

the plaintiff's case citation focusing on the legal factor disclosure_in_negotiations is not adequate for deciding the new case c_{new}.

5 Discussion

Also in Hypo [1]. hypothetical variations of a current case are created. Hypo's variations are created by adding some legal factors to the current case. It should be noted that cases in Hypo is a legal case in our sense. As illustrated in the previous section. our creation of hypothetical similar raw case often yields a hypothetical legal case including some additional legal factors. Therefore, it would be considered that our creation of hypotheticals is closely related to Hypo's one. However, their purposes are quite different. In Hypo. the purpose of creating hypotheticals is to show how certain variations could strengthen or weaken legal arguments for a side. More concretely, Hype creates hypotheticals in order to show which legal factors could strengthen or weaken the side. On the other hand, our creation of hypotheticals is for the examination of the stability of opponent's case citation. We emphasize here that in our creation of hypotheticals, some additional legal factors are yielded *side-effectively*, while in Hypo's one, some legal factors are added by Hypo itself *intentionally*.

Our knowledge-base in this paper is assumed to be a set of Horn-rules in which no negations are appeared. However, allowing negations in knowledge-base would be desired to represent more practical legal knowledge. As well known, we have two types of negations in the first-order logic, *classical (logical) negation* and *negation as failure (NAF)*. The current GDA would be able to deal with a knowledge-base with classical negations by a slight modification, since the underlying Tenenberg's abstraction framework has been proposed for such a knowledge-base [2]. In order to deal with NAF, however. further investigation on GDA should be made. It has currently been under studying.

6 Concluding Remarks

In this paper. we formalized a computational method for attacking a legal argument by refuting its case citation. Although several ways of argument attack have been investigated. we newly proposed to attack from the viewpoint of stability of case citation. Since our basic idea of stability is based on a reasonable requirement. the authors expect that the notion of stability would widely be accepted. In order to examine stability of case citation. we create a hypothetical similar cases with the help of GDA. Our GDA is an useful algorithm for finding a similarity depending on a viewpoint. We would be able to use GDA in many legal reasoning systems. Since similarity plays a very important role in legal domain. some mechanism for computationally dealing with similarity is highly desired to construct useful systems. GDA can be used as the basis of such a powerful mechanism. Some applications of GDA to analogical legal reasoning has been investigated in [3. 4].

It should be emphasized that although this paper investigated a method for attacking an argument from the viewpoint of the stability of its case citation, use of the stability is not limited to that purpose. It can be used in general as a criterion for case citation in Case-Based Reasoning systems. That is, when we have several candidates of case citation, we prefer a stable one to the others.

The work in this paper is the first step in constructing a legal argumentation system with GDA. Since our method was investigated in a restricted Hypo-like system, integrating the method into the original Hypo would be desired as the next step. By such an integration, we believe strongly that Hypo would become a more powerful and useful case-based legal argumentation system.

Acknowledgments

The authors would like to thank anonymous referees for their very helpful comments and suggestions.

References

[1] Kevin D. Ashley, "Modeling Legal Argument : Reasoning with Cases and Hypotheticals", The MIT Press, 1990.

[2] Josh D. Tenenberg, "Abstracting First-Order Theories", Change of Representation and Inductive Bias (D.Paul Benjamin, ed.), Kluwer Academic Publishers, pp.67–79, 1989.

[3] T. Kakuta, M. Haraguchi and Y. Okubo, "A Goal-Dependent Abstraction for Legal Reasoning by Analogy", Artificial Intelligence and Law, Kluwer Academic Publishers, 1997 (in printing).

[4] T. Kakuta. M. Haraguchi and Y. Okubo, "Legal Reasoning by Structural Analogy Based on Goal-Dependent Abstraction", Proceedings of the Ninth International Conference on Legal Knowledge-Based Systems – JURIX'96, Tilburg University Press, pp.111–121, 1996.

[5] Y. Okubo and M. Haraguchi. "Constructing Predicate Mappings for Goal-Dependent Abstraction", Proceedings of the Fifth International Workshop on Algorithmic Learning Theory (ALT'94), Lecture Note in Artificial Intelligence 872. Springer-Verlag, pp.516-531, 1994.