Abductive Reasoning for Burden of Proof

Ken Satoh[1], Satoshi Tojo[2], Yoshitaka Suzuki[2]

[1] National Institute of Informatics and Sokendai
ksatoh@nii.ac.jp
[2] Japan Advanced Institute of Science and Technology
{tojo, syoshita}@jaist.ac.jp

Abstract. In argumentation system, it is important to reason about winning strategy during argument according to the context of their counterpart arguments and their own past arguments. In legal reasoning, it is more complicated than the ordinary argumentation system, since it involves a notion of a *burden of proof*.
In this paper, we extend our framework to reason about the *next move* given context of arguments in legal reasoning by introducing an *abductive framework* into our previous work [7] of a formalization of burden of proof. We show that an abductive framework is useful for reasoning about the next suitable measure in order to win a case.

1 Introduction

In [7], we gave a formalization of burden of proof and showed that a switch of burden of proof (SBP) can be expressed correctly in the nonmonotonic formalization without any additional functions which Prakken used in [4]. We use an interpretation of burden of proof in the Japanese civil procedure law as follows. In the Japanese civil procedure law, a burden of proof is a tool for decision making by a judge when an ultimate fact is "non liquet", that is, the state which we cannot conclude that the ultimate fact is true or not even after giving all available evidence [8]. This burden is decided a priori in a general way and there is no influence of context of individual cases. We formalize the status of "non liquet" as the status in which judges cannot decide its truth value. When an ultimate fact is "non liquet", the default value of the fact is used and as a side-effect, if some default value is beneficiary to one side, then the other side must prove the negation of the default value to prevent the status of "non liquet".

We explain a burden of proof by the following tort case.

- Suppose that plaintiff rents his house to defendant and defendant subleases the house to his sister for her small business without permission of plaintiff.
- Then, plaintiff has the burden of proving that there was the above fact (sublease) in order for plaintiff to cancel the contract of house rental by Article 612 of the Japanese civil law; (1) A lessee may not assign the lessee's rights or sublease a leased thing without obtaining the approval of the lessor and (2) If the lessee allows any third party to make use of or take the profits of a leased thing in violation of the provisions of the preceding paragraph, the lessor may cancel the contract.

This means that if the truth-value of the fact "sublease" is not decided (that is, in "non liquet" status) yet at the final judgement after providing full evidence, we assume that the truth value of the fact is false and therefore, plaintiff will lose this case. Therefore, plaintiff needs to strictly prove the fact "sublease" by giving an evidence of sublease and rebutting a contradictory evidence given by defendant.

Then, a switch of burden of a proof occurs as follows.

- Suppose that defendant concedes plaintiff's claims concerning sublease, and instead attacks plaintiff's argument by claiming an exception, viz. that her small business does not harm the contract of house rental. This kind of defense by non-faith-sublease is allowed by the case law in Japan [9].
- In this case, defendant has the burden of proving non-bad-faith sublease, for example, by claiming that her small business is just two hours lesson of weaving once a week and therefore does not harm the house in any way.

Therefore, in this situation an allocation of proof is switched to defendant and as far as the issue of defendant's non-bad-faith sublease is concerned, if there is no evidence from defendant or there are contradictory pieces of evidence for non-bad-faith, then the truth value of non-bad-faith sublease is false and defendant will lose this case.

Prakken claims that in existing argumentation systems, we cannot formalize a switch of burden of proof since this switch is not automatically decided in such a way that burden of proof is switched alternately. Moreover, Prakken claims that nonmonotonic reasoning cannot formalize a switch of burden of proof since the argumentation system can represent any nonmonotonic reasoning.

We [7] challenged this claim by formalizing a switch of burden of proof as simulation of a decision making of judge in court in stead of formalizing a switch of burden of proof as a turn change in dialogue game. In our work, a switch of burden is not a dynamic process during argumentation, but a tool for decision process after all the argument are already done. Considering this decision process in advance, every side should give evidence and argumentation in order to get a preferred status in the process. Therefore, our formalization is quite different from existing argumentation systems since in our formalism every side makes arguments as if there were a third party who makes judgement according to these arguments.

In legal reasoning including a burden of proof, there are two kinds of arguments:

- one of which we call a *strict proof* where the side making a strict proof must provide evidence to prove its proposition and rebut all the counter-argument about the proof, and
- the other of which we call a *counter-argument* of a proof where the side making a counter-argument must provide a counter-evidence which contradicts the evidence to prove the proposition.

If one side has a burden of proof of a proposition, then the side must provide a strict proof of the proposition and it is sufficient for the other side to provide a

counter-argument in order to invalidate the proof. So, each side should find out which fact they should provide according to the above two kinds of arguments.

Moreover, there is a critical legal problem to specify which fact is an ultimate fact to decide the case. In the above example of sublease, defendant must prove the "non-bad-faith" sublease. However, there are two theories which we call *standard-as-ultimate theory* and *fact-as-ultimate theory* to specify the way of proving the "non-bad-faith" as follows.

Standard-as-ultimate theory: In this theory, ultimate fact is considered to be a conceptual standard term such as "non-bad-faith". Then, defendant has a burden of alleging "non-bad-faith" directly and provides evidential facts which support the ultimate fact, "non-bad-faith". The provided facts are not ultimate facts but evidences which the judge could use to decide whether "non-bad-faith" is satisfied in the current situation. Moreover, the judge could reason about "non-bad-faith" by using not only the evidential facts which defendant provides but also ones which plaintiff carelessly provides.

Fact-as-ultimate theory: In this theory, ultimate facts are considered to be concrete facts which supports "non-bad-faith". Then, defendant has a burden of alleging the concrete facts. The judge decides whether these concrete facts are satisfied in the current situation and then evaluate these concrete facts mean "non-bad-faith". In this case, the judge can not use other facts which are not alleged by defendant.

So, choosing appropriate one among these theories affects difficulty of proving facts and therefore, each side needs to argue about which theory should be used.

As another example related with a decision on the burden of proof, there would be several theories which side should have the burden of proof. In tort cases, plaintiff usually has a burden of proof to prove defendant's fault. However, in traffic accident cases in Japan, because of difficulty of proving the fault, the burden of proof is shifted to defendant. That is, in traffic accident cases, defendant must prove the nonexistence of his faults in order for him to win the case. This kind of *shift of burden of proof* could occur in a type of cases where the main evidence exists in the side of defendants and is hard for plaintiff to obtain.

Therefore, we need to take into account of these theoretical arguments to decide the next move and sometimes before providing the evidence, we could argue which fact should be proved as an ultimate fact and also which side has the burden of proof. We show that using abduction we can infer not only the next move but also reason about which fact is an ultimate fact and which side has the burden of proof.

2 Formalizing Burden of Proof by Logic Programming

We first review our previous framework of a formalization of burden of proof in [7] [3] . In the formalism, we introduce a predicate `p` and the meaning of `p(P)`

[3] We modify the previous formalization by introducing the notion of the degree of proof.

for an ultimate fact `P` is that in the current status of provided evidence, the ultimate fact `P` is above the degree of being proved (we sometimes say that `P` is above the degree of proof). In this paper, for simplicity, we separate reasoning using ultimate facts from decision of the (three-valued) truth value of ultimate fact `P` given a set of evidence. Therefore, we start reasoning after we decide whether `p(P)` is either true, false or unknown.

A strict proof of an ultimate fact which is used for ordinary proof is expressed using a predicate `proved` as follows:

```
proved(P) :- p(P).
```

Suppose that the default value of `P` is true in "non liquet" status. Then, this fact becomes true either by giving evidence proving `P` or by making `P` "non liquet" status. To distinguish between a fact which requires a strict proof and a fact which can use the default value, we introduce another predicate `default` to express this situation and we have the following rule by using the predicate `proved` as follows:

```
default(P) :- p(P).
default(P) :- not p(P), not p(-P).
```

where

- `-P` means a explicit negation of `P`.
- `not` of `not p(P)` represents "negation as failure" which means the failure of providing evidence of `p(P)`.
- `not p(P), not p(-P)` means "non liquet" status since in the current status of evidence, neither `P` nor `-P` is above the degree of proof.

The above rules are simplified as follows.

```
default(P) :- not proved(-P).
```

Since it is impossible to hold `p(P)` and `p(-P)` simultaneously, `p(P)` is equivalent to `p(P)` and `not p(-P)`. Therefore using the above two rules of `default`, we get `not p(-P)` as the definition of `default` which is equivalent to `not proved(-P)`.

This means that if this default value is beneficiary to one side, the other side must prove the negation of the default value strictly. In this case, we can say that the burden of proof of proving the negation of `P` is allocated to the opponent.

In a general case, we make the following rule. For any conclusion, we divide ultimate facts used to prove conclusions into the two categories; one of them should be proved by the proponent (we represent as `P_1,...,P_n`) and the negation of the other of them should be proved by the opponent (we present these conditions as `-O_1,...,-O_m`) to rebut the conclusion. Then, we have the following rule:

```
proved(C) :-
    proved(P_1), proved(P_2),..., proved(P_n),
    default(O_1),...,default(O_m).
```

In the above formalization, which side has the burden of proof is not so clear. We can express who has the burden of proof by adding arguments for each predicate as follows. In the following program, the second argument in the `p`-predicate expresses the side which has the burden of proof of the first argument and the third argument in `default` expresses the side which has the burden of proof of the negation of the first argument.

So, we now have the following rule:

```
proved(C,P,O) :-
    proved(P_1,P,O), proved(P_2,P,O),..., proved(P_n,P,O),
    default(O_1,P,O),...,default(O_m,P,O).
proved(X,P,O):- p(X,P,O).
default(X,P,O):- inverse(X,InX),not proved(InX,O,P).
inverse(-X,X).
inverse(X,-X):- atomic(X).
```

where

- In the body of predicate `default`, we exchange the argument place of the proponent expressed as `P` and the opponent expressed as `O` in the call of the predicate `proved`. This corresponds with a switch of burden of proof.
- We introduce the predicate `inverse` since we will consider both positive and negative atoms.

The example in Section 1 can be formalized in the following logic program. Note that the burden of proof is already decided in advance as stated in Section 1, therefore we can write the following rules beforehand.

```
proved(cancel_contract,P,O):-
    theory(standard_as_ultimate),
    proved(sublease,P,O),
    default(-non_bad_faith,P,O).
proved(sublease,P,O):- p(sublease,P,O).
proved(non_bad_faith,P,O):- p(non_bad_faith,P,O).
proved(cancel_contract,P,O):-
    theory(fact_as_ultimate),
    proved(sublease,P,O),
    default(-fact_evaluated_as_non_bad_faith,P,O).
proved(fact_evaluated_as_non_bad_faith,P,O):-
    p(fact_evaluated_as_non_bad_faith,P,O),
    default(-fact_disevaluated_as_non_bad_faith,P,O).
proved(fact_disevaluated_as_non_bad_faith,P,O):-
    p(fact_disevaluated_as_non_bad_faith,P,O).
default(X,P,O):-inverse(X,InX), not proved(InX,O,P).
```

- The first rule,

```
proved(cancel_contract,P,O):-
```

```
theory(standard_as_ultimate),
proved(sublease,P,O),
default(-non_bad_faith,P,O).
```

means that to cancel the contract of house rental, the proponent has to show that there is a sublease and there is no ultimate fact representing a non-bad-faith sublease. This rule comes from *standard-as-ultimate theory*. Therefore, we introduce one more predicate `theory` to specify which theory is used. Then, to rebut this rule, the opponent must prove that the sublease is a no-bad-faith sublease. This is done by showing `proved(non_bad_faith)` in the last rule.

– On the other hand, if we assume *fact-as-ultimate theory*, we must use the fourth rule:

```
proved(cancel_contract,P,O):-
    theory(fact_as_ultimate),
    proved(sublease,P,O),
    default(-fact_evaluated_as_non_bad_faith,P,O).
```

where we must provide some fact as an ultimate fact which is evaluated as non-bad-faith sublease. In order to defend against the above argument, the opponent must prove an ultimate fact which is evaluated as a non-bad-faith sublease as in the fifth rule:

```
proved(fact_evaluated_as_non_bad_faith,P,O):-
    p(fact_evaluated_as_non_bad_faith,P,O),
    default(-fact_disevaluated_as_non_bad_faith,P,O).
```

But in this case, there is a defense from the opponent proving an ultimate fact which is disevaluated as a non-bad-faith sublease.

3 Reasoning about the Next Move by Abductive Logic Programming

In this section, we extend our framework in order to reason about the *next move* by abduction.

Firstly, we give a definition of abductive framework and its semantics [2].

Definition 1. *An* abductive framework *is a pair* $\langle T, A\rangle$ *where* A *is a set of predicate symbols, called* abducible predicates *and* T *is a set of rules each of whose head is not in* A*.*

We call a set of all ground atoms for predicates in A *abducibles*.

Definition 2. *Let* T *be a logic program and* Π_T *be a set of ground rules obtained by replacing all variables in each rule 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 (possibly infinite) program.*

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

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* `false` $\notin M$.

Definition 3. *Let* $\langle T, A\rangle$ *be an abductive framework and* Θ *be a set of abducibles. A* generalized stable model $M(\Theta)$ *w.r.t.* $\langle T, A\rangle$ *is a stable model of* $T \cup \{H \leftarrow | H \in \Theta\}$.

We say that A generalized stable model $M(\Theta)$ *has a minimal set of abducibles if there exists no strict subset* Θ' *of* Θ *s.t.* $M(\Theta')$ *is a generalized stable model.*

We give a translation method from the above logic program into an abductive logic program as follows. There are two situations; in one situation, plaintiff will lose a case and plaintiff would like to win a case by providing further evidence from the plaintiff. In the other situation, defendant will lose a case and defendant would like to win a case by providing further evidence from the defendant

In order for plaintiff to win the case, we do the following.

1. We introduce abducible predicate `p*` and add the following rules:

   ```
   p(C,plaintiff,defendant) :- p*(C,plaintiff,defendant).
   false :- p(C,P,O), p*(-C,P,O).
   false :- p(-C,P,O), p*(C,P,O).
   ```

   where `p*(C,plaintiff,defendant)` means that plaintiff must provide some evidence in order for the judge to decide that `C` is true. The last two rules are used to prevent contradictory assumptions.
2. We introduce abducible predicate `ce*` and add the following rules:

   ```
   p(C,defendant,plaintiff) :- not ce*(C,plaintiff,defendant).
   ```

   for `C` which is currently shown to be above the degree of proof by defendant. In the above rule, `ce*(C,plaintiff,defendant)` means that plaintiff must provide some counter-evidence for `C` which leads to "non liquet" status for `C`. Note that in our framework, we only reason about the need of giving counter-evidence.
3. We introduce abducible predicate `theory*` and add the following rule:

   ```
   theory(T) :- theory*(T).
   ```

   This abducible is used to assume a theory which is preferable to the plaintiff. We may have integrity constraints to avoid introducing theories which contradict each other.

In order for defendant to win the case, we do a similar thing to the above where we exchange `plaintiff` and `defendant` in the above definition.

By asking `proved(C,plaintiff,defendant)` for a top query where C is a conclusion of the case, a generalized stable model with a minimal set of abducibles gives the suitable next move for plaintiff while by asking `not proved(C,plaintiff,defendant)`, a generalized stable model which has a minimal set of abducibles gives the suitable next move for defendant.

4 Example

We now give examples to reason about the next move. We use the same example used in Section 2[4] .

Example 1. Firstly, suppose that there is no evidence given from either side. Then, plaintiff will lose a case. To reason about the next move in order for plaintiff to win, we add the following rules to the rules in Section 2.

```
p(s,p,d) :- p*(s,p,d).
p(nbf,p,d) :- p*(nbf,p,d).
p(fenbf,p,d) :- p*(fenbf,p,d).
p(fdenbf,p,d) :- p*(fdenbf,p,d).
theory(su) :- theory*(su).
theory(fu) :- theory*(fu).
false :- theory*(su), theory*(fu).
false :- not theory*(su), not theory*(fu).
```

Note that the last two rules enforce judge to enforce one of these theories. Then, from this program, if we ask `proved(cancel_contract,p,d)`, we obtain
$\Theta 1 = \{$ `theory*(su),p*(s,p,d)`$\}$ and $\Theta 2 = \{$ `theory*(fu),p*(s,p,d)`$\}$ as minimal sets of abducibles. This means that if plaintiff would like to win, plaintiff must provide evidence of sublease and it does not matter which theory for ultimate facts is used.

Example 2. Secondly, suppose that there is enough evidence to prove sublease. In this context, defendant will lose the case. To reason about the next move in order for defendant to win, we provide the following rules. (Note that the argument places of `p` and `d` are exchanged.)

```
p(s,p,d):- not ce*(s,d,p).
p(s,d,p) :- p*(s,d,p).
p(nbf,d,p) :- p*(nbf,d,p).
p(fenbf,d,p) :- p*(fenbf,d,p).
p(fdenbf,d,p) :- p*(fdenbf,d,p).
theory(su) :- theory*(su).
theory(fu) :- theory*(fu).
false :- theory*(su), theory*(fu).
false :- not theory*(su), not theory*(fu).
```

Then, if we ask `not proved(cancel_contract,p,d)` for the program, we obtain the following minimal sets of abducibles.
$\Theta_3 = \{$`ce*(s,d,p),theory*(su)`$\}$
$\Theta_4 = \{$`ce*(s,d,p),theory*(fu)`$\}$
$\Theta_5 = \{$`theory*(su),p*(nbf,d,p)`$\}$

[4] We abbreviate `plaintiff` as `p`, `defedant` as `d`, `sublease` as `s`, `non_bad_faith` as `nbf`, `fact_evaluated_as_non_bad_faith` as `fenbf`, `fact_disevaluated_as_non_bad_faith` as `fdenbf`, `standard_as_ultimate` as `su`, and `fact_as_ultimate` as `fu`.

$\Theta_6 = \{$`theory*(fu),p*(fenbf,d,p)`$\}$

Θ_3 and Θ_4 mean that if defendant gives counter-evidence for sublease, then defendant will win no matter which theory for ultimate fact is used. Θ_5 means that if defendant takes the standard-as-ultimate theory, then defendant must show non-bad-faith whereas Θ_6 means that if defendant takes the fact-as-ultimate theory, then defendant must show a fact which is evaluated as non-bad-faith sublease.

Example 3. Finally, suppose that defendant takes the standard-as-ultimate theory and gives sufficient evidence of non-bad-faith sublease. This also means that defendant provides a fact which is evaluated as non-bad-faith sublease even if defendant takes the fact-as-ultimate theory. Note that plaintiff can argue that they should take the fact-as-ultimate theory even if defendant takes the other theory. To reason about the next move in order for plaintiff to win, we provide the following rules.

```
p(s,p,d).
p(nbf,d,p) :- not ce*(nbf,p,d).
p(s,p,d) :- p*(s,p,d).
p(nbf,p,d) :- p*(nbf,p,d).
p(fenbf,p,d) :- p*(fenbf,p,d).
p(fenbf,d,p) :- not ce*(fenbf,p,d).
p(fdenbf,p,d) :- p*(fdenbf,p,d).
theory(su) :- theory*(su).
theory(fu) :- theory*(fu).
false :- theory*(su), theory*(fu).
false :- not theory*(su), not theory*(fu).
```

Then, if we ask `proved(cancel_contract,p,d)` for the program, we obtain the following minimal sets of abducibles.

$\Theta_7 = \{$`theory*(su),ce*(nbf,d,p)`$\}$

$\Theta_8 = \{$`theory*(fu),ce*(fenbf,p,d)`$\}$

$\Theta_9 = \{$`theory*(fu),p*(fdenbf,p,d)`$\}$

Θ_7 means that if plaintiff takes the standard-as-ultimate theory then plaintiff needs to provide a counter-evidence for non-bad-faith sublease. On the other hand if plaintiff takes the fact-as-ultimate theory then plaintiff needs to provide a counter-evidence for the fact provided by defendant which is evaluated as non-bad-faith sublease (Θ_8) or needs to provide a fact which is disevaluated as non-bad-faith sublease (Θ_9).

5 Conclusion

There has been research which uses abduction in legal reasoning [1, 4, 6]. [1] suggests usage of Poole's abductive system [3] to find out a counter-argument in legal argumentation. [4] uses abduction to reason about evidence in causal reasoning in legal case. In [6], we use minimal abduction to give a solution to

the paradox of *conditio sine qua non*. However, none of these work considers the *next move* problem of argumentation systems.

In this paper, we have extended our formalization of burden of proof to reason about the next move for a side to win given context using abductive logic programming. For the future work, we should verify utility of our framework for more practical examples and also extend our framework to manipulate "burden of production" proposed by Prakken and Sartor[5].

Acknowledgments We thank anonymous referees very much for their giving constructive comments on this paper.

References

1. Gordon, T., F., Issue Spotting in a System for Searching Interpretation Spaces, *Proc. of ICAIL 1989*, pp. 157 – 164 (1989).
2. Kakas, A. C., Kowalski, R., Toni, F., The Role of Abduction in Logic Programming , *In: Handbook of Logic in Artificial Intelligence and Logic Programming 5, D.M. Gabbay, C.J. Hogger and J.A. Robinson eds., Oxford University Press* pp. 235 – 324 (1998).
3. Poole, D., A Logical Framework for Default Reasoning, *Artificial Intelligence*, Vol 36, pp. 27 – 47 (1988).
4. Prakken, H., Modelling Defeasibility in Law: Logic or Procedure?, *Fundam. Inform.*, 48(2-3), pp. 253-271 (2001).
5. Prakken, H., Sartor, G., Formalising arguments about the burden of persuasion, Prof. of ICAIL 2007, pp. 97 –106
6. Satoh, K., Tojo, S., "Disjunction of Causes and Disjunctive Cause: a Solution to the Paradox of 'Conditio Sine Qua Non' using Minimal Abduction", Legal Knowledge and Information Systems, JURIX 2006: The Nineteenth Annual Conference, pp. 163 – 168 (2006).
7. Satoh, K., Tojo, S., Suzuki, Y., "Formalizing a Switch of Burden of Proof by Logic Programming", International Workshop on Juris-Informatics (JURISIN 2007), pp. 76 – 85 (2007).
8. Shindo, K., New Civil Procedure Law (Revised 3rd Edition), Kobundo publisher (in Japanese) (2005).
9. Supreme Court Judgement, 1966, Jan. 27., Minshu., Vol. 20, No. 1, p. 136-144 (in Japanese)(1966).