Interactive System for Arranging Issues based on PROLEG in Civil Litigation

Ken Satoh
National Institute of Informatics
Chiyoda, Tokyo, Japan
ksatoh@nii.ac.jp

Kazuko Takahashi
Kwansei Gakuin University
Sanda, Hyogo, Japan
ktaka@kwansei.ac.jp

Tatsuki Kawasaki
educe Co.,Ltd
Chiyoda, Tokyo, Japan
sktk40829@gmail.com

1 INTRODUCTION

In Japan, we have the procedure of "arranging issues" in civil litigation where we clarify which facts are in dispute and what kind of evidence action should be made for these issues. Currently, IT technology is used only for online meeting for arranging issues and more sophisticated method is expected by a full use of IT/AI technology.

We proposed a method for formalizing Japanese presupposed Ultimate Fact theory (JUF theory, in short, Youken-jijisturon, in Japanese) and converting it into logic programming and developed a system called PROLEG (PROlog-based LEGal reasoning support system)[1]. JUF formalises which party should give certain facts to get a desired legal effect for the party, in other words, formalizes which party has a burden of proof for these facts. Then, given these facts, PROLEG simulates reasoning by a judge to make a final conclusion and present such process in a directed tree structure called "block diagram."

In this work, we modify the PROLEG system to support arranging issues in civil litigation. In the PROLEG system, they assume that all the facts are given before simulation of judge's reasoning so there are no interaction between both parties of plaintiff and defendant during the simulation. On the other hand, given a desired effect requested by one party, our interactive system (which we call *int-PROLEG*) automatically calculates possible justifications for the desired effect based on JUF theory stored in the system. After the party chooses a justification and int-PROLEG asks for the existence of necessary facts to the party to satisfy the justification. Then, int-PROLEG asks whether the other party agrees on alleged facts and also calculates possible counter-arguments against the chosen justification and provides them to the other party. We iterate this process until no further (counter-)arguments are presented. When this process is finished, disagreed facts are issues for which a judge decides the truth value.

Most interactive argument systems are mainly for constructing arguments manually by a user or evaluating arguments constructed by a user. A notable exception would be the Carneades system which has a funcition of argument invention using argumentation schemes [2]. On the other hand, in our system, a user does not construct arguments from the scratch but chooses a pattern of legal arguments provided by int-PROLEG based on the JUF theory and specifies concrete facts to make concrete legal arguments for a specific case in a civil litigation. Moreover, from our experience working with lawyers, we noticed that if we introduced a system with sophisticated but complex reasoning mechanisms, they would be very reluctant to use the system. So, our purpose of this work is to identify the simplest function to arrange issues which would be easily understood by lawyers. In a sense, we extract a useful part of Carneades for arrangement of issues in civil litigation.

2 PROLEG

We firstly review the PROLEG system[1]. A program of PROLEG consists of a *rulebase* and a *factbase.* A rulebase consists of a set of *general rules* of the form

$$H \Leftarrow B_1, ..., B_n.$$

where H (called *head* or *conclusion*), $B_1, ..., B_n$ (called *body*) are first-order atom, and a set of *exception rules* of the form

$$exception(H, E)$$

where H and E are the head of some general rules. We call E *exception.* A factbase consists of set of the following expression $fact(P)$ where P is an atom which is never the head of any general rule. We call P *fact predicate.*

A rule represents a general default rule meaning that if B_i in R are all proved then *in general* H is true except there is an exception rule $exception(H, E)$ such that E is proved.

Given a PROLEG program, we can construct a proof tree of a given goal which is the root of the tree and the child nodes are conditions of general rules of the conclusion and the exceptions of the conclusion.

3 EXTENSION TO ARRANGE ISSUES

In this section, we show how to modify the PROLEG system into int-PROLEG.

3.1 Indexing a level for PROLEG literals

(1) First, we define the dependency on the atomic formula that appears in the general rule. Among the conclusions of the general rule, the conclusion that does not appear in the body of any general rule or is not an exception of any exception rule is called 0-*level conclusion.* Then, when making a top-down proof tree from the 0-level conclusion using only the general rules, we end up the fact predicates. We call the fact predicates finally visited 0-*level facts* and the 0-level conclusion and the intermediate visited atomic formulas called 0-*level atomic formulas.*

(2) Suppose that the i-level atomic formulas and the i-level facts are decided. For exception rules that conclude with the i-level atomic formula, the collection of the atomic formulas

ICAIL'21, June 21–25, 2021, São Paulo, Brazil

of the exceptions of the such exception rules are called $(i+1)$*-level exception rules*. When making a top-down proof tree from the $(i+1)$-level exception using only the general rules, we end up the atomic formulas. We call the fact predicates finally visited $(i+1)$*-level facts* and the $(i+1)$-level exception and the intermediate visited atomic formulas called $(i+1)$-*level atomic formulas*.

3.2 Interaction Process in int-PROLEG

Let D be a defendant and P be a plaintiff.

(1) Let P choose one of the following 0-level conclusion. This is P's claim.
(2) We make a top-down proof tree from 0-level conclusion and when we encounter the 0-level fact, we ask P if the P claims that fact. If P claims it, then we let P to instantiate variables in 0-level facts and added instantiated facts to the fact base and we reflect the relevant part of the proof tree with such instantiation. If P does not claim it, we delete the path related with the fact in a proof tree.
(3) If all the concrete facts are entered, the modified proof tree is displayed.
(4) Suppose that a user finished to enter t-level facts. We make a proof tree for $(t+1)$-level exception and when we encounter the $(t+1)$-level fact,
   - if $(t+1)$ is odd, we ask D if D claims that fact. If D claims it, then we let D to instantiate variables in $(t+1)$-level facts and added instantiated facts to the fact base and we reflect the relevant part of the proof tree with such instantiation. If D does not claim it, we delete the path related with the fact in a proof tree. We also ask D whether D admits $turn$-level fact or not, and if D does not admit it, we make a label of the fact as an "issue to be determined".
   - if $(t+1)$ is even, we ask P if P claims that fact. If P claims it, then we let the P to instantiate variables in $(t+1)$-level facts and added instantiated facts to the fact base and we reflect the relevant part of the proof tree with such instantiation. If P does not claim it, we delete the path related with the fact in a proof tree. We also ask P whether P admits $turn$-level fact or not, and if P does not admit it, we make a label of the fact as an "issue to be determined".
(5) If all the concrete facts are entered, the modified proof tree is displayed.
(6) Continue above until there are no new argument raised.

The final proof tree will be all the arguments raised by the both parties and fact nodes labelled as an "issue to be determined" are issues for which a judge decide the truth value.

4 DEMONSTRATION

We show a demonstration how a plaintiff and a defendant interact each other to arrange issues in the following claims about claiming payment in a purchase contract[1]. In the example case, the plaintiff (Alice) requested the defendant (Bob) for the payment of goods according to the purchase contract with Bob. Bob claimed that Alice threatened Bob to buy the goods for a counter-argument, but Alice did not admit the threatening action. The system works as follows.

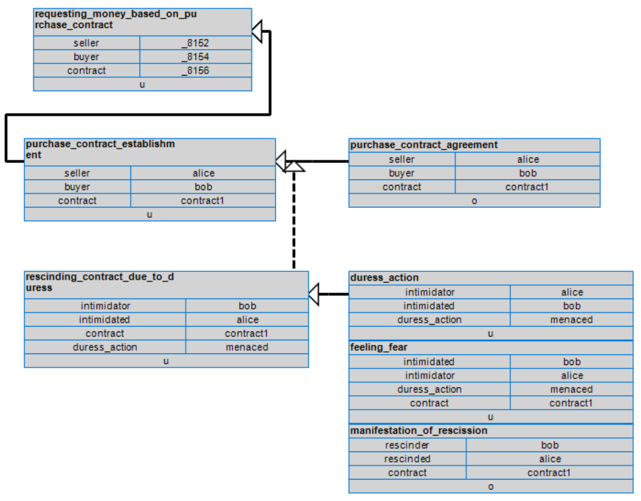

Figure 1: Final Diagram

(1) Firstly, Alice claims the payment and the system tells a necessary fact to establish the claim according to the Civil Code, which is an establishment of a purchase contract. Alice gives a specific fact for the purchase contract (called "contract1").
(2) Secondly, the system computes counter arguments against Alice's claim according to the Civil Code and provides two counter arguments ("already paid" argument and "menaced to establish the contract" argument). Bob chooses "menaced to establish the contract" argument and gives relevant fact for the argument. (A block conntected with a dotted arrow in Fig.1 represents a counter argument.).
(3) Thirdly, the system computes counter arguments agaist Bob's claim but founds no counter arguments. So the only thing Alice can do is to deny some facts. In this example, Alice denies the facts related with Bob's claim of being menaced so these facts become issues for which a judge decides the truth value. (Leaf blocks with 'u' in the bottom in Fig.1 are issues.).

5 CONCLUSION

We present how to extend the PROLEG system to arrange issues. This system helps for lawyers to prevent missing claims and for lay person to make correct legal claims without lawyers. For a future research, we need an evaluation of the system and investigate a method of automatic extraction of relevant legal facts directly from claims in natural language.

Acknowledgements: This work was supported by JSPS KAKENHI, JP17H06103 and JP19H05470 and JST AIP, JPMJCR20G4.

REFERENCES

[1] K. Satoh et al. 2012. PROLEG: An Implementation of the Presupposed Ultimate Fact Theory of Japanese Civil Code by PROLOG Technology. In *New Frontiers in Artificial Intelligence: JSAI-isAI 2010 Workshops, Revised Selected Papers, LNAI 6797*. 153–164.
[2] D. Walton and T. F. Gordon. 2017. Argument Invention with the Carneades Argumentation System. *Scripted* 14, 2 (2017), 168–207. https://script-ed.org/?p=3391

[1]http://research.nii.ac.jp/~ksatoh/PROLEGdemo/IssueArrangmentDemo.mp4