PROLEG: Practical Legal Reasoning System

Ken Satoh(✉)

National Institute of Informatics, 2-1-2, Hitotsubashi, Chiyoda-ku, Tokyo, Japan
ksatoh@nii.ac.jp

Abstract. This paper introduces a legal knowledge representation language, PROLEG. PROLEG rules are general rules in the form of Horn clauses and special meta-prediate expressing exceptions. Exceptions are introduced to express negative information in stead of "negation as failure". It is because reasoning pattern of general rules and exceptions fits lawyers' reasoning and therefore lawyers understand PROLEG easily. We firstly give the definition of syntax and semantics of PROLEG and show an application for legal reasoning.

Keywords: PROLEG · Logic Programming · Legal Reasoning

1 Background

After many years of theoretical research on logic programming and nonmotonic reasoning, I sought practical applications of my theoretical research and I entered law school in 2006 and learned "Japanese presupposed ultimate fact theory" (we write "JUF theory" for short in this paper) which was developed in lawyers training center in Japan at the law school. This theory is to help judges to make reasonable conclusions even under incomplete information environment due to lack of evidence. I immediately understood the aim of this theory is exactly same as nonmonotonic reasoning and am sure that I can implement the reasoning in JUF theory [10].

My understanding of JUF theory is as follows:
In a litigation, the truth values of some facts which contribute to the judgement might be unknown due to the sufficient evidence. Then, from the deductive reasoning, the correct logical condition for the judgment is "unknown" as well. However, judges are not allowed to give such unknown judgement but have to give decisive answer. To solve this problem, JUF theory attach a default truth value to every condition in Japanese Civil Code and let judges use the default value when the condition is unknown in the litigation. Then all the conditions are determined by real truth values or default truth values and therefore conclude the decisive judgement. Actually, an attached default value is closely related with the burden of proof. Since if the default value is favorable to the plaintiff (the defendant, respectively), the defendant (the plaintiff, respectively) must prove the negation of the default value otherwise the defendant (the plaintiff, respectively) lose the litigation.

2 PROLEG

We firstly started to write legal rules in PROLOG [10] reflecting burden of proof in a similar way to the British Nationality Act in PROLOG [13]. However, we found that lawyers have difficulty to understand negation as failure in PROLOG so we changed the syntax from negation as failure into a framework of general rules and exceptions which is a common reasoning pattern among lawyers to create PROLEG (PROlog based LEGal reasoning support system) [7].

Our main aim is to make lawyers to use legal reasoning system by providing a minimum legal language sufficient for the reasoning so that lawyers understand the behavior of the system. Our approach is quite opposite with academic trends in AI and law in that researchers introduce many subtlty to express detailed deontic modality. As far as pratical legal system is concerned, however, the legal systems which usual AI and Law researchers provide is too complicated for lawyers who do not have a background of logic and thus the lawyers do not use them.

Although we have not conducted any psychological experiments, we had experiences on PROLEG with law school graduates who write PROLEG solution for Japanese bar exams for each year from 2009 to 2022 (total more than 60 graduates) in that they can start to make a program in PROLEG after a few weeks training of programming in PROLEG. I believe that PROLEG is the most familiar legal knowledge representation language for lawyers and has a potential to be a de fact standard.

Now, we introduce PROLEG. PROLEG system consists of a rulebase and a fact base.

- A PROLEG rulebase consists of the following expression.
  - A rule of the form of Horn clauses (without negation as failure):
    $$H \Leftarrow B1, ..., Bn.$$
  - An exception is an expression of the form:
    $$exception(H, E).$$
    where H, E are atoms each of which is the head of a rule.
- A PROLEG factbase consists of the truth value of related facts in a case. We use an expression for a fact P in a case as:
  $$fact(P).$$

The intuitive meaning of PROLEG rules is that if the conditions of general rules are satisfied, its conclusion is satisfied in general but if the exception E is satisfied the conclusion is no longer true. Note that E of $exception(H, E)$ is the head of a rule so there is a general rule whose head is E. Then we could write an exception E of exception E' by representing as $exception(E, E')$.

The semantics of PROLEG program is defined as follows [7]. We make a program to be grounded by the constants in the program and name it as P.

Let M be a set of atoms. We define a set of applicable rules w.r.t. M, P^M, as follows:

$$\{R \in P | \text{there is no } E \text{ s.t. } exception(head(R), E) \text{ and } E \in M\}$$

This means that if some exception is found for a conclusion H of rule R, we do not allow such a rule R to participate in a derivation. The semantics of P (called an extension of P) is given as a set of atoms M s.t. $M = min(P^M)$ where $min(T)$ is the minimum model of T.

It is analogous to answer set definition and actually, PROLOG and PROLEG is mathematically equivalent in the sense that there is a one-to-one translation from PROLEG to PROLOG and vice versa. Here, we reproduce the equivalence translation according to [9].

Suppose that we have a program whose general rules are as follows:
$C \Leftarrow B_{11}, ..., B_{1n_1}$.
$C \Leftarrow B_{21}, ..., B_{2n_2}$.
$\vdots$
$C \Leftarrow B_{k1}, ..., B_{kn_k}$.
and excetions are as follows:
$exception(C, E_1)$.
$\vdots$
$exception(C, E_m)$.

Then, we can traslate the above PROLEG program into the following program:
$C : -B_{11}, ..., B_{1n_1}$, `not` $E_1, ...,$ `not` E_m.
C: $-B_{21}, ..., B_{2n_2}$, `not` $E_1, ...,$ `not` E_m.
$\vdots$
C: $-B_{k1}, ..., B_{kn_k}$, `not` $E_1, ...,$ `not` E_m.
Note that rules with the same head has the same negative literals. If we add some facts in PROLEG and PROLOG, we can show that derived literals are equivalent.

On the other hand, suppose that we have the following PROLOG program:
C: $-B_{11}, ..., B_{1n_1}$, `not` $E_{11}, ...,$ `not` E_{1m_1}.
C: $-B_{21}, ..., B_{2n_2}$, `not` $E_{21}, ...,$ `not` E_{2m_1}.
$\vdots$
C: $-B_{k1}, ..., B_{kn_k}$, `not` $E_{k1}, ...,$ `not` E_{km_k}.
Then, we can translate a PROLOG program into the following PROLEG program using additional predicate C_i.
$C \Leftarrow C_1$. $C_1 \Leftarrow B_{11}, ..., B_{1n_1}$.
$C \Leftarrow C_2$. $C_2 \Leftarrow B_{21}, ..., B_{2n_2}$.
$\vdots$
$C \Leftarrow C_k$. $C_k \Leftarrow B_{k1}, ..., B_{kn_k}$.
$exception(C_1, E_{11})$. $\cdots$ $exception(C_1, E_{1m_1})$.

$exception(C_2, E_{21}).$ $\cdots$ $exception(C_2, E_{2m_2}).$

$\vdots$

$exception(C_k, E_{k1}).$ $\cdots$ $exception(C_k, E_{km_k}).$

If we add some facts in PROLEG and PROLOG, we can show that derived literals except additional predicate C_i's are equivalent.

It is interesting that even two languages are mathematically equivalent but understandability of lawyers is different.

Moreover, this way of writing rules explicitly reflects a burden of proof in litigation. The burden of proof for the conditions of a general rule resides in the party who wants its conclusion to be satisfied whereas the burden of proof for exceptions resides in the party who wants to deny the conclusion. Therefore, it is useful for lawyers in civil litigation to decide which evidence should be collected to win the case. Here is an example of PROLEG rules in contract law. We omit detailed arguments in each predicate for the sake of explanation.

```
right_to_ask_payment(Seller,Buyer,Object,Price)<=
    purchase_contract_establishment( Seller,Buyer,Object,Price).
% A seller has a right to force a buyer to make an payment
% over the object if a purchase contract is established.

purchase_contract_establishment(Seller,Buyer,Object,Price)<=
    purchase_agreement(Seller,Buyer,Object,Price).
% A purchase contract is established if there is an agreement
% of purchase of the object.

exception(right_to_ask_payment(Seller,Buyer,Object,Price),
          payment(Buyer,Seller,Object,Price)).
% There is an exception about sellerÂĄfs right to ask payment
% if buyer made payment.

payment(Buyer,Seller,Object,Price)<=
    payment_fact(Buyer,Seller,Object,Price).
% Payment is made if there is a fact of payment.
```

And here is a case description using PROLEG facts

```
fact(purchase_agreement(bob,alice,television,1000  euro)).
% Bob sold the television from Alice at the price of 1000 euro.

fact(payment_fact(alice,bob,television,1000 euro)).
% Alice paid 1000 euro to Bob for television.
```

PROLEG provides an explanation of the reasoning process to a judgement using a block diagram. We show a block diagram for the above case in Fig. 1. The explanation of block diagram is as follows:

– Right-handside top-most block expresses a judgement.

– A bottom item of each block expresses the result of evaluation of conclusions/conditions; o: success, x: fail
– A solid line between blocks expresses conclusion-condition relation for a general rule.
– A dotted line shows exception of the conclusion of a general rule.

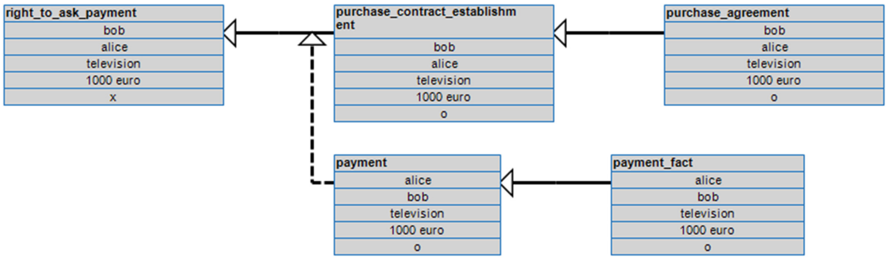

Fig. 1. PROLEG block diagram

3 Current Status of PROLEG and Its Applicability and Possible Extensions

We have been constructing a large rule base of 2500 rules and exceptions consisting of civil code and supreme court case rules since 2009. As far as we know, it is the largest legal rule base in the world. We checked the correctness of the rulebase to solve the multiple-choice part of Japanese bar exams by the law school graduates from University of Tokyo which is one of the best law school in Japan for 2009–2022. To manage such a large database, debugging tools are essentially necessary so we have investigated such legal debugging [1] as well. Regarding the efficiency of legal reasoning in civil litigation, a judge must provide a decisive decision so the structure of rule base in PROLEG is a stratified logic programming to guarantee a binary decision so that efficient implementation is possible. As a direct application of PROLEG in civil code litigation, PROLEG can be used to check a missing arguments which should be made by lawyers and as an educational tool, for law school students to enhance their understanding on legal reasoning based on burden of proof. We extend PROLEG into an interactive system to arrange issues [11][1]. We have been developing ODR (Online Dispute Reasoning) system more intelligent than E-Bay ODR system by enhancing man/machine interface of PROLEG system [6]. We have also been investigating a combination of natural language processing and PROLEG so that a lay user can write a case description in natural language and NLP

[1] You can see the demo video at http://research.nii.ac.jp/~ksatoh/PROLEGdemo/IssueArrangmentDemo.mp4.

extracts necessary information for an input to PROLEG so that a lay user can find out an outcome of his/her problem by PROLEG block diagram [2–4].

There are various directions for extension. Since PROLEG is a framework to write legal rules in terms of general rules and exceptions so we could formalize statutory laws in general [9]. We have investigated an application to criminal law [5], GDPR [12] and an application to Private International Law [8].

4 Conclusion

We have described our activities of logic programming paradigm in legal domain and explained our legal knowledge representation language PROLEG, which has a lot of potential for supporting various legal activities. We strongly believe that PROLEG would be one of the prominent practical application in logic programming.

Acknowledgements. This research was supported by JSPS KAKENHI Grant Numbers, JP17H06103, JP19H05470 and JP22H00543, and JST, AIP Trilateral AI Research, Grant Number JPMJCR20G4, Japan.

References

1. Fungwacharakorn, W., Tsushima, K., Satoh, K.: On the legal debugging in PROLEG program. In: Advances in Intelligent Systems and Computing, vol. 1357, pp. 25–36 (2021). https://doi.org/10.1007/978-3-030-73113-7_3
2. Navas-Loro, M., Satoh, K., Rodríguez-Doncel, V.: ContractFrames: bridging the gap between natural language and logics in contract law. In: Kojima, K., Sakamoto, M., Mineshima, K., Satoh, K. (eds.) JSAI-isAI 2018. LNCS (LNAI), vol. 11717, pp. 101–114. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-31605-1_9
3. Nguyen, H.T., Fungwacharakorn, W., Nishino, F., Satoh, K.: A multi-step approach in translating natural language into logical formulas. In: Proceedings of the 35th International Conference on Legal Knowledge and Information Systems (JURIX 2022), pp. 103–112 (2022). https://doi.org/10.3233/FAIA220453
4. Nguyen, H.T., Nishino, F., Fujita, M., Satoh, K.: An interactive natural language interface for PROLEG. In: Proceedings of the 35th International Conference on Legal Knowledge and Information Systems (JURIX 2022), pp. 294–297 (2022). https://doi.org/10.3233/faia220484
5. Nishigai, Y., Satoh, K.: Programming of "Japanese presupposed ultimate fact theory" in criminal law using PROLEG (in Japanese). Inf. Network Law Rev. **19**, 81–120 (2021). https://doi.org/10.34374/inlaw.19.0_81
6. Nishioka, S., Satoh, K., Mori, Y.: Consumer dispute resolution system based on PROLEG. In: Proceedings of the 35th International Conference on Legal Knowledge and Information Systems (JURIX 2022), pp. 298–301 (2022). https://doi.org/10.3233/FAIA220485
7. Satoh, K., et al.: PROLEG: an implementation of the presupposed ultimate fact theory of Japanese civil code by PROLOG technology. In: Onada, T., Bekki, D., McCready, E. (eds.) JSAI-isAI 2010. LNCS (LNAI), vol. 6797, pp. 153–164. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25655-4_14

8. Satoh, K., Giordano, L., Baldoni, M.: Implementation of choice of jurisdiction and law in private international law by PROLEG meta-interpreter. In: Baroni, P., Benzmüller, C., Wáng, Y.N. (eds.) CLAR 2021. LNCS (LNAI), vol. 13040, pp. 60–75. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-89391-0_4
9. Satoh, K., Kogawa, T., Okada, N., Omori, K., Omura, S., Tsuchiya, K.: On generality of PROLEG knowledge representation. In: Proceedings of the 6th International Workshop on Juris-informatics (JURISIN 2012), pp. 115–128 (2012a)
10. Satoh, K., Kubota, M., Nishigai, Y., Takano, C.: Translating the Japanese presupposed ultimate fact theory into logic programming. In: Proceedings of the 22nd Annual Conference on Legal Knowledge and Information Systems (JURIX 2009), pp. 162–171 (2009). https://doi.org/10.3233/978-1-60750-082-7-162
11. Satoh, K., Takahashi, K., Kawasaki, T.: Interactive system for arranging issues based on PROLEG in civil litigation. In: Proceedings of the Eighteenth International Conference on Artificial Intelligence and Law (ICAIL 2021), pp. 273–274 (2021). https://doi.org/10.1145/3462757.3466096
12. Sawasaki, T., Troussel, A., Satoh, K.: A use case on GDPR of Modular-PROLEG for private international law. In: Proceedings of the 3th International Workshop on Artificial Intelligence Technologies for Legal Documents (AI4LEGAL 2022), pp. 1–11 (2022)
13. Sergot, M.J., Sadri, F., Kowalski, R.A., Kriwaczek, F., Hammond, P., Cory, H.T.: The british nationality act as a logic program. Commun. ACM **29**(5), 370–386 (1986). https://doi.org/10.1145/5689.5920