Representation of Legal Knowledge by Compound Predicate Formula

Hajime Yoshino*

1 Introduction

The essential points in developing any method to represent legal knowledge are that: (1) the method is easy for lawyers to understand and use, (2) it has the sufficient ability to express legal knowledge in detail, and (3) it is applicable to formalizing legal reasoning. For these tree reasons I have designed Compound Predicate Formulas (CPF), which is a conservative extension of the first order predicate logic, as a method of the representation of legal knowledge and developed Legal Expert Systems (LES-2 and LES-3) on its basis. In this paper I explain the method, illustrating some examples, and also give the rigorous logical foundation of CPF i.e. the establishment of its syntax and semantics.

2 Why CPF ?

In this section I will explain the reason why I have introduced CPF, not others. To say the reason in brief phrases, in order to represent legal knowledge adequately and plainly. To clarify this point, we will take a simple example from legal sentences and present the difficulty of representing such a sentence by the standard first order logic. Consider this:

Ex. John made an offer to Mary, and it was accepted.

It is difficult to express the whole sentence in the standard first order language, for the standard first order language does not contain any device for representing the referential expression "it". We can symbolize, in the above sentence, "John made an offer to Mary" as "offer (John, Mary)," but how can we symbolize "it was accepted"? We all agree that the referential pronoun "it" that is part of the above sentence refers "an offer of John to Mary." Unfortunately the first order language has the ability to refer only individual entities but not any state of affair such as an offer made of John to Mary. As far as we are in the standard first order language, we must content ourselves with symbolizing the above example like, using a predicate $p(X_1, X_2)$, $offer(John, Mary)$. Though, this symbolizing does not reflect the inner structure this sentence has. This fact implies that the standard first order language is not rich enough to adequately represent legal sentences, and therefore to describe legal reasoning.

*Meiji Gakuin University: 1-2-37, shirokanedai, minato-ku, Tokyo, Japan; e-mail: hyoshino@tansei.u-tokyo.ac.jp.

In short, the standard first order language lacks the means to refer to each legal act, which involves "that contract" or "the trade at 15:00 on Feb. 3rd, 1994." Moreover the standard language has no device to represent referential pronouns, say "that." What we really need is some richer language that enables us to deal with these expressions.

3 Some Precedent Approach

3.1 Lambda Abstraction

For instance, if the sentence "X contracts with Y" is formalized as "contract," then the relation of contracting is , according to lambda notation, described as:

$$\lambda X \lambda Y(contract(X, Y))$$

In general given a predicate, lambda operators form the expression that refer to the concept the predicate denotes. However, by this lambda abstraction it is very difficult to designate any particular contract e.g. "the contract made between John and Mary at that time." Again what we would like to obtain is the method for referring each concrete instantiation of given legal relations rather than one for any abstract concept.

3.2 Class Notation

Then what if the class notation ? In this notation, relations corresponds to classes and each instance, for example:

$$A = \{X : acceptance(X)\}$$

$$O = \{Z : \exists X \exists Y(offer(X,Y) \wedge Z =< X,Y >\}$$

$$\exists Z(Z \in O \wedge Z \in A)$$

A and O show (the extension of)a concept of "acceptance" and a concept "offer," respectively. Then the below (1) is obtained as the translation of a legal sentence "X made an offer to Y and it was accepted":

(1) $\exists Z \exists X \exists Y(offer(X,Y) \wedge Z =< X,Y > \wedge acceptance(Z))$

As (1) shows, the class notation is adequate for denoting a concept itself while this method is clearly not satisfactory for denoting its particular instance. Indeed each instance is a certain element of a given class, as already noted. Many of expressions in natural languages, however, involve ample pragmatical i.e. contextual information, and it is hard to specify a purposed set, considering much contextual information, and even though the specification is acquired, the formula is all too often complex for us to understand. Also the class notation does not have any type of apparatus to represent referential pronouns. In legal sentences and legal reasoning, one does meet with referential expressions frequently.

4 A New Device–ID-symbols

In the previous section we have recognized that some approaches to overcoming difficulties as to the standard first order language are inadequate to express each individual legal act or to describe the sentences with referentials in the form which reflects the inner structure of them. Now we are in a stage to offer a new device for coping with these puzzles: ID-symbols. Though, before introducing ID-symbols we will account for the rough idea of them from which they stem.

Let us suppose that:

1. $offer(Z, X, Y)$:Z is an offer of X to Y.
2. $acceptance(W, Z)$:W is the acceptance of Z.

If we assume these formulas, then a sentence "An offer of X to Y was accepted." would be formalized as follows:

(2) $\exists Z(offer(X, Y, Z) \wedge acceptance(Z, W))$

Compared (2) with (1), we find that (2) is more simple and plain, for in the assumptions expressions within which they contain the way for referring to a certain specified individual legal act or relation had been posited. And this is how we contrive a device called ID-symbols.

In general for any predicate $p(X_1, \ldots, X_n)$,

$$ID - p(X_1, \ldots, X_n)$$

is the predicate in question. For instance, for a predicate "contract(Mary,John),"

$$ID - contract(Mary, John)$$

expresses a contract between Mary and John. In other words, it is the name of that contract. Using ID-symbols, the formula (2) is more concisely rewritten as

(3) $acceptance(-, ID - offer, -)$

Here I would like to emphasize, as the characteristics of ID-symbols, this sort of nominalization, i.e. an ID-symbol forms the name of a particular instance in a given concept. (Recall that notations by lambda operators or classes form the name of a concept itself.) As one more example, "the reject of that offer" is, by ID-symbols, formalized as

(4) $ID - reject(-, ID - offer(-), -)$

Using ID-symbols, we can easily deal with such a relatively complex case.

We may note, in passing, that, for ID-symbols to function as names, it is necessary that the obvious identity criterion for the referents of Id-symbols is given. It means that for particular instances of a concept the condition of continuity through time must be defined. So as to define that condition, we ought to define each legal concept strictly. But this problem is the matter of law and not that of logic.

5 Outline of Syntax of CPF

The following attempts to define the syntax of CPF. Most portions are the same as the standard first order language, CPF is highly different from the language so far in that CPF has new devices such as case symbols[1] and ID-symbols. We have to define an describe the syntactical behaviors of ID-symbols more fully than here. But our concern here restricted only to program clauses. So we will think only quantifier free part i.e. Horn clause:

$$B \longleftarrow A_1, \ldots A_n$$

where $B, A_1, \ldots, A_n$ are literals.

The syntax of CPF is as follows:

1.Basic Vocabulary:

1.1.individual variables: $X_1, X_2, \ldots T_1, T_2, \ldots$

1.2.individual constants: $a_1, a_2, \ldots$

1.3.case symbols $agt :, obj :, goa :, tim :, \ldots$

1.4.predicate letters: $p_1, p_2, \ldots$

1.5.list symbols [,]

1.6.logical constants: $\neg, \longleftarrow, \forall$

1.7.commas, parentheses: ( ) ,

2.terms and formulas:

2.1. Variables, individual constants and ID-symbols are terms.

2.2. If t is an individual constant, an individual variable or an ID-symbol, then $agt : t$, $obj : t$, $tim : t$, $goa : t$ are terms. ($c_1, c_2, \ldots$stand for case symbols.)

2.3. $[t_1, \ldots, t_n](t_i(1 \leq i \leq n))$ is a list.

2.4. $p([t_1, \ldots, t_n])$ is a formula.

2.5. If A and B are formulas, then $\neg A, A \longleftarrow B$ are formulas.

2.6. If $A(X)$ is a formula, then $\forall X A(X)$ is a formula.

2.7.The definition of ID-symbols: For the predicate representing legal concept $p([t_1, \ldots, t_n])$, $ID - p([t_1, \ldots, t_n])$ is an ID-symbols of its predicate.[2]

[1] Case symbols are a notation contrived to express the inner structure of predicate explicitly. On the syntactic and semantic status of case symbols, we may leave to another occasion.

[2] We will omit the arguments in ID-symbols unless leading to misunderstanding. Derivatively we will define ID-symbols about predicate symbols as well.

2.8. For a predicate $p([t_1,\ldots,t_n])$, $p(ID-p,[t_1,\ldots,t_n])$ is a formula as well.[3]

2.9. An expression is a formula only if it can be shown to be a formula on the basis of conditions 2.4-2.6, 2.8.

Other logical constants are introduced by the definitions below:

$$A \wedge B \triangleq \neg(\neg B \longleftarrow A)$$

$$A \vee B \triangleq B \longleftarrow \neg A$$

$$\exists X A X \triangleq \neg(\forall X \neg A X)$$

6 Legal Knowledge Representation in terms of CPF

Having defined syntax of CPF, we state legal knowledge representation using CPF. Here we cite an article of United Nations Convention for the International Sale of Goods (CISG) and show how it can be translated into a formula of the language of CPF.

CISG article 23: A contract is concluded when an acceptance of an offer becomes effective.

1. $contract(ID-co,[agt:[X,Y],obj:C])$: A contract C was made between X and Y.
2. $acceptance(ID-ac,[agt:X,obj:ID-of,goa:Y])$: X accepted $ID-of$ to Y
3. $offer(ID-of,[agt:X,goa:Y,obj:C])$: An offer C was made to Y by X.
4. $be-concluded(ID-bc,[obj:ID-co,tim:T])$: $ID-co$ was concluded at time T.
5. $become-effective(ID-be,[obj:ID-ac,tim:T])$: $ID-ac$ became effective at time T.

Based on these, the above article is translated into the formula below.

$$\begin{aligned}&be-concluded(ID-bc,[obj:ID-co,tim:T_1])\\ &\quad \wedge contract(ID-co,[agt:[X,Y],obj:C]) \longleftarrow\\ &\quad become-effective(ID-be,[obj:ID-ac,tim:T_1])\\ &\quad \wedge acceptance(ID-ac,[obj:ID-of])\\ &\quad \wedge offer(ID-of,[agt:X,goa:Y,obj:C])\end{aligned}$$

Such a formula is called Flattized CPF formula(FCPF), and it is equivalent to the CPF formula below:

$$\begin{aligned}&be-concluded(ID-bc,[obj:contract(ID-co,[agt:[X,Y],obj:C]),tim:T_1]) \longleftarrow\\ &\quad become-effective(ID-be,[obj:acceptance(ID-ac,[agt:X,goa:Y,\\ &\quad obj:offer(ID-of,[agt:X,goa:Y,obj:C])])],tim:T_1])\end{aligned}$$

[3] The former is called the formula without ID-symbol, the latter the formula with ID-symbol as a matter of convenience. As easily seen, we need a formula to assure $p(t_1,\ldots,t_n) \Longleftrightarrow p(ID-p,t_1,\ldots,t_n)$,if we make an axiomatic system for a CPF.

This formula is an abbreviation of the above FCPF formula. Legal sentences are described and stored into knowledge base in this form. To execute the predicational reasoning, these formulas are compiled (flattized) into FCPF above.[4]

Next is the outline of procedure of the flattization. Any CPF formula A is flattized into an FCPF formula, i.e., for any CPF formula A,

1. if A contains no formulas which have the form of $p(ID-p,[c_1:t_1,\ldots,c_i:q(ID-q,[]),\ldots,c_n:t_n](1\le i\le n)$ in A , the formula is not flattized.
2. if A contains any formulas described in 1, choose the left-most one in that formulas, replace $c:q(ID-q,[\,])$ with $c:ID-q$, and replace the original formula with the below one,

$$p(ID-p,[\ldots,c:ID-q,\ldots]) \wedge q(ID-q,[])$$

3. Repeat the procedure of 2 until it is not applicable.

7 Application of CPF to Legal Reasoning

In CPF ID-symbols play an important role. We have already seen some advantages of ID-symbols. In this section I will expand an advantage by the introduction of ID-symbols in legal reasoning. Since our CPF has its basis on the standard first order language, we can use inference rules of it. Besides that, ID-symbols increased the power of our language so that we could deal with some legal reasoning cases that have been difficult to cope with so far. For example, let us consider an inference like this:

Premise1. A made a contract with B.

Premise2. If that contract is effective, then A can claim to payment to B.

Premise3. That contract is effective.

Conclusion. A can claim payment to B.

We will be in trouble with this inference if we have to formalize this within the standard first order language. The trouble is derived from that the standard predicate logic has no device for referring to any particular instance like " that contract ". On the other hand, CPF tells us that the above inference is valid. The formalization by CPF is below:

Premise1'. $contract(ID-co,A,B)$

Premise2'. $claim-payment(A,B) \longleftarrow is-effective(ID-ie,ID-co)$

Premise3'. $is-effective(ID-ie,ID-co)$

Conclusion'. $claim-payment(A,B)$

[4] This flattization is, substantially, the procedure of converting a many-sorted formula into a one-sorted one. If we applied the order sorted logic, we could deal with formulas of CPF directly.

Notice that we can deal with this inference not because we have extended the inference rules of the standard first order logic (in fact we have not extended them), but because we have introduced ID-symbols, which enables us to refer to particular instances of a given act. That is to say, in the case of quantifier-free part, CPFL is an extension of the standard first order language.

I would like to suggest more two points about the usage of ID-symbols: in an inference (1) even if a particular ID-symbol is used with its argument not specified, we may identify the ID-symbol safely, and (2) when a particular ID-symbol is used and embedded in another ID-symbol such as a case , $ID - r(t_1, \ldots, ID - contract, \ldots, t_n)$, we might be in trouble as to the identification of given plural ID-symbols.

8 Semantics of CPF

We can define semantics of CPF as usual. Only difference between usual first order language and that of CPF is the introduction of ID-symbols in the latter.

The definition of a model of CPF is as follows.

Definition 8.0.1 (Level of ID-symbols) *Given an ID-symbol $ID - P(t_1, \ldots, t_n)$, the number of ID-symbols in $ID - P(t_1, \ldots, t_n)$ is called a level of the ID-symbol, and we express it as $LEVEL(ID - P(t_1, \ldots, t_n))$.*

Definition 8.0.2 *$M =< D(= D_1 \cup D_2 \cup \{\bot\}), I >$ is a model of CPFL with respect to $g(= g_1 \cup g_2) \iff$*

1. *D_1 and D_2 are a class of individuals and a class of time points respectively. We stipulate that $D_1 \cap D_2 = \emptyset$. $\bot \notin D_1 \cup D_2$.*
2. *$g_1 : INDVAR^5 \longmapsto D_1$ and $g_2 : TIMVAR \longmapsto D_2$.*
3. *If t is an individual constant, $I(t) \in D_1$.*
4. *Given an n-place predicate p, $I(p) \subseteq D^n$ where D^n is a Cartesian product of D with n times.*

5.**interpretation of ID-symbols** *Given a predicate $p(t_1, \ldots, t_n)$, the interpretation of its ID-symbol $I_g(ID - p(t_1, \ldots, t_n))$ is defined by the induction on the level of the ID-symbol.[6] $I(ID - p)$ is a function defined on D.[7] If $LEVEL((ID - p(t_1, \ldots, t_n))) = n \geq 2$, and the interpretation of ID-symbols of the level less than n have been defined, then $I_g(ID - p(t_1, \ldots, t_n))$ is defined as follows:*

   1. *If $(I(ID - p) \neq \emptyset$, pick up an a such that $a \in I(ID - p)$ so that $I_g(ID - p(t_1, \ldots, t_n)) = a$.*
   2. *Otherwise, $I_g(ID - p((t_1), \ldots, (t_n))) = \bot$.*

[5] $INDVAR$, $TIMVAR$ are the set of the individual variables and the set of the time variables respectively.

[6] I_g is a function such that for a variable X, $I_g(X) = g(X)$ and for a term t of the other kind, $I_g(t) = I(t)$.

[7] In a special case, $I(ID - p)$ is a function. As to the reason why we have defined in a more general way, we will explain later. The meaning of an ID-symbol is substantially identical to the meaning of constant in the case of quantifier-free formulas.

Definition 8.0.3 (Satisfaction of the Formulas) *We define the satisfaction of the formulas of CPF by the induction on the complexity of them. Namely, for any model $M =< D, I >$ and any assignment g,*

1. $M \models_g p(t_1, \ldots, t_n) \iff < I_g(t_1), \ldots, I_g(t_n) > \in I(p)$[8]

1.1. $M \models_g p(ID - p, t_1, \ldots, t_n) \iff < I_g(t_1), \ldots, I_g(t_n) > \in I(p)$[9]

2. $M \models_g \neg A \iff M \not\models_g A$[10]
3. $M \models_g B \longleftarrow A \iff M \not\models_g A$ or $M \models_g B$
4. $M \models_g \forall X A(X) \iff \forall g' s.t. g' =_X g, (M \models_{g'} A(X))$[11].

We should explain the idea behind the model construction above. Interpretation of ID-symbols needs special explanation. If we understand an ID-symbol by analogy to a function,

$$ID - r :< X_1, \ldots, X_n > \longmapsto X_{n+1}$$

Namely,

$$ID - r(X_1, \ldots, X_n) = X_{n+1}$$

Therefore, we are tempted to define as follows: given an assignment g,

$$I_g(ID - r(X_1, \ldots, X_n)) = I_g(ID - r)(< g(X_1), \ldots, g(X_n) >)$$

where $I_g(ID - r)$ is a function.

If $I(ID - r)$ is a function with non-empty range, its corresponding role of $I(ID - r)$ in ordinary language is that of a singular term. We have expressions of this kind. For example, "that contract between a and b"is such an expression. But in any case, can we say that the above mentioned contract is unique or many or none? If there are several contract between A and B, and we can't determine the special one by the lack of information, then the decision remains obscure. Suppose that there are several contracts between A and B such as the contract on March 17, and the contract on November 16. For this reason, it is possible that when debating, they misunderstand each other what contract they are talking about. And it is also possible that when we infer about some contract, we do so without complete knowledge of the contract. Rather it seems that such a reasoning is typical in our ordinary life.

We would like to comment on the interpretation of ID-symbols.

- If we don't have enough information of ID-r to make it function, then $I(ID-r)$ would be a correspondence. But if it is a function, then there will be no problem of misidentification. In this case we can think with appropriate objects.

[8] $M \models_g A$ should be read as: g satisfies A in M. In CPF predicate of the original form, every argument of the predicate appear within a list, but for the sake of convenience, we employ predicates in a standard form. There is no essential difference between them.

[9] The satisfaction is defined in the same way for atomic formula without ID-symbols $p(t_1, \ldots, t_n)$ and atomic formula with ID-symbols $p(ID - p, t_1, \ldots, t_n)$.

[10] $M \not\models_g A$ should be read as: g does not satisfy A in M.

[11] $g =_X g' \overset{\Delta}{\iff}$ for any Y s.t. $Y \neq X g(Y) = g'(Y)$

9 Conclusion

We have so far been stating the quantifier-free part of CPF (syntax, legal knowledge representation using it, semantics, and so on). At the end we are going to summarize in short merits of introducing CPF, especially ID-symbol.

- CPF makes expressive capacity richer, and can mention each legal acts and relations.
- CPF makes legal knowledge representation which has close form to natural language.

In this way CPF has big advantages. Above all the best significance of this paper is to give a logical basis of CPF.

I would like to state further tasks. I have hesitated offering the explanation of case symbols in order to avoid making obscure the forms of argument, but they are an important tool. Case symbols are a device for clarifying that which role terms play in a predicate. It is interesting to give semantics to such a category of grammar.

I then have often mentioned the characteristics of ID-symbols as a demonstrative pronoun. Next steps, we must consider other kinds of demonstrative pronoun (indexicals such as "I", demonstratives such as "the book I have" and so on) from the wider point of view. Now I present two points to consider in future.

- How to formally identify objects that are represented by making some extension of first order language
- How to combine ID-symbols and many sorted language

10 References

[1] Ebbinghaus, H.D., J.Flum and W.Thomas, *Mathematical Logic*, Springer,1984.

[2] Gupta, A., *The Logic of Common Noun*, Yale University Press, 1980.

[3] Rödig. J, Über die Notwendigkeit einer besonderen Logik der Normen, in: Albert, H. et al. (hrsg.) *Rechtstheorie als Grundlagenwissenschaft der Rechtwissenschaft, Jahrbuch für Rechtssoziologie und Rechtstheorie* Bd,2 ,1972, S. 163-185

[4] Sakurai, Seichiro and Yoshino, Hajime, Identification of Implicit Legal Requirements with Legal Abstract Knowledge, in: *Proc. The Fourth International Conference on Artificial Intelligence and Law*, ACM (The Association for Computing Machinery), 1993, pp. 298-305

[5] Smullyan, R., *First-order Logic*, Springer,1968.

[6] Yoshino, Hajime, Über die Notwendigkeit einer besonderen Normenlogik als Methode der juristischen Logik, in: Klug, U. et al. (hrsg.), *Gesetzgebungstheorie, Jurisitische Logik, Zivil- und Prozeßrecht (Gedächtnisschricht für Jürgen Rödig)*, Springer-Verlag Berlin Heidelberg, 1978, S. 140-161

[7] Yoshino, Hajime, Zur Anwendbarkeit der Regeln der Logik auf Rechtsnormen, in: *Die Reine Rechtslehre in wissenschaftlicher Diskussion (Schriftenreihe des Hans Kelsen-Instituts Band 7)*, Wien(Manz-Verlag), S. 142-164

[8] Yoshino, Hajime, Logical Structure of Law and the Possibility of Computer Aided Legal Reasoning, in: *ARSP(Archiev für Rechts- und Sozialphilosophie)* Beihefte Nr.30, 1986, pp. 185-202

[9] Yoshino, Hajime, et al., Legal Expert System Les-2, in: Wada, E. (Ed.), *Logic Programming '86 (Lecture Notes in Computer Science 264)*, 1987, p. 36ff.

[10] Yoshino, Hajime, *Research Report on Legal Expert System*, Association for Machine System Promotion, 1989, pp.51-81 (in Japanese)

[11] Yoshino, Hajime, *Research Report on Explication of Legal Knowledge Structure and Development of Legal Knowledge-base*, Association for Machine System Promotion, 1990, pp.27-32, pp.41-55 (in Japanese)

[12] Yoshino, Hajime et al., Towards a Legal Analogical Reasoning System Knowledge Representation and Reasoning Methods, in: *Proc. The Fourth International Conference on Artificial Intelligence and Law*, ACM (Association for Computing Machinery), 1993, pp. 110-116