II. After having decided to stay in line with the prototype or to divert from it, it would only be natural to lay down as well, in which direction one has decided to divert or not to divert. I feel that up to now one has designed databases for decisions too much like databases for literature. A court decision on theft, however, is something very different from an article on the legal history of theft. It is always a decision pro and contra. A decision pro theft is at the same time a decision contra fraud, or contra robbery, or contra embezzlement, or contra the taking of a corpse (think of a mummy), or against a case of mere furtum usus, which as a rule is not punishable.

From the standpoint of computer science, the following is trivial, but nevertheless worth remembering from time to time: If I search my database for decisions on theft *and not* fraud, I will not get decisions pro theft and contra fraud, but those decisions that contain the word 'theft' and not the word 'fraud'. So I have to search for theft *and* fraud, but now I will receive all decisions where both the words 'theft' and 'fraud' are mentioned, among them many I will not need.

A retrieval system suitable for decisions can be easily obtained by splitting the column for solutions dichotomically in 'pro' and 'contra' or 'ascribed category' and 'denied category'. Now it is easy to find all decisions in which theft is marked off from fraud or, for that, from robbery. Contrary to the form of topoi dichotomisation we began with, the dichotomisation of solutions is not inclusive, but exclusive (pro/contra).

The essential question is whether it is possible to make this distinction easily, quickly, and with intersubjective evidence. This is the case especially as far as higher court instances are concerned. This follows from the mechanism of legal appealing. We have a lower court decision and an appeal against it. As a rule the higher court will confirm either the previous decision or the appeal. Since a court decision has to be enforcable, it should be possible to extract its gist with certainty and from the first pages of the document. By dichotomising the field 'solution', one only revokes an unnecessary abandonment of information.

[illegible]

LES-Project

Hajime Yoshino / Munenori Kithahara

1. Introduction

Expert systems are enthusiastically being developed in many fields. Progress in the field of law has made less headway than in other fields, for example, medicine and engineering. The popularity of Prolog these days has, however, served to promote legal expert systems in Japan. For example, in the fields of inheritance taxes law, patent law, copyright law, contract law and so on, expert systems have been developed. In November 1984, the *Legal Expert System Association* (abbr. LESA) was organized by Prof. *Hajime Yoshino*, the members consisting of specialists of different fields including law, logic, theory of language, information science, and knowledge engineering. Since then LESA has gone far toward developing legal expert systems in Japan.

An expert system is a computer system (a software) containing specified knowledge, with which one can perform a problem-solving task. When an expert system is built in the field of law, it is expected to support the lawyer's work, and to serve as legal consulting, legal study, and legal education auxiliary instruments. The problem-solving work is, in the main part, called legal reasoning. Therefore, an expert system in the legal field must be a legal reasoning system in this sense.

In this paper, we first briefly discuss the present status of development of legal expert systems in Japan. In order to build a legal expert system, it is necessary to accumulate legal knowledge and represent it appropriately in the system. Then it is important that the knowledge used in the system should be written down, that is, recorded in the knowledge base, according to its characteristics. Therefore, in section 3, we clarify the characteristics we think of - they are really reflected in our building of the system. And finally we outline our legal expert system, LES-2.

2. The Present Status of Development of Legal Expert Systems in Japan

In this section, we intend to sketch the present condition of development of legal expert systems in Japan.

In 1983, Mr. *Ikeda*, an attorney-at-law, developed an inheritance taxes law expert system on a personal computer, on an NEC PC9800 employing Prolog/KABA, for study of legal professionals. The system now runs on VAX11/785, transplanted in QUINTUS-Prolog by Japan DEC. This system was the first expert system developed by a lawyer. This system is a reasoning system determining legal relations by using directly the function of reasoning and backward reasoning of Prolog.

Dr. *Nitta*, a knowledge engineer in the Electrotechnical Laboratory, developed a legal expert system for patent law in 1983. Patent law is a complicated law composed of a substantial law and a procedural one. For a layman, the area is very hard to understand. *Nitta* developed a knowledge representation and inference system for procedural law (KRIP/L) based on Prolog in order to express the legal concepts, for example, *withdrawal, nullity and annulment of defects of procedures*, etc., which involve concepts of time. By integrating object-oriented class concepts and section logic, the system is able to describe the complicated relations between substantial law and procedural law. This system in part contains patent law and the related field in civil law as legal knowledge, and has been equipped with two functions, namely the retrieval of laws and the adaptability to case problems. At the beginning, the system ran on a personal computer NEC PC9800 by Prolog/KABA, but it is now being run on a PSI (high speed Prolog machine) at ICOT (Institute for New Generation Computer Technology).

In 1984 an expert system for copyright law was developed by Prof. *H. Tanaka* and Mr. T. *Ikeda* (Tokyo Institute of Technology). *Tanaka* and *Ikeda* developed the DCKR (Definite Clause Knowledge Representation), a remarkable knowledge representation method. DCKR represents knowledge by using a Horn clause of Prolog. The inheritance of knowledge, the knowledge representation of if-need type, and the inference are dealt with in the built-in function of Prolog. In this system, legal concepts are classified into four types. They are: *right*, *act*, *object* and

agent. The relation between these four concepts is described by a DCKR. DCKR employs *sem* as a fundamental predicate, and one-to-one correspondence between Horn clause and slot. If the first arguments of the sem predicate of the heads are the same, they are represented as one structured object (or frame). By this method, legal rules are written in sem predicates by matching one Horn clause to one legal article.

Here, we introduce two expert systems built by a construction company. These systems do not now belong to a legal reasoning system, but contain construction acts.

Construction Acts Consultation System by the Ohbayashigumi Construction Company contains the knowledge base of the building standard act and the acts concerned. One of the characteristics of the standard acts is numerical control. At the planning stage of consultation, a planner (i.e., user) can get an answer within the control. They further intend to develop this CAD system.

The Land Use Consulting System developed by Tokyu Construction Company employs the Land Use Consulting System for acts concerned, which have been authorized by the government in land use. Therefore, the system deals with 13 acts, such as the building standard act, the forest act, town planning and zoning acts, and local regulations.

Before these legal expert systems were developed, Prof. *Yoshino* developed a reasoning system in contract law (i.e., the *A-Project*). In this system, first, contract law is systematically analyzed by propositional calculus. This system infers a conclusion following the logic flow charts written in BASIC and assembler, and runs on an NEC PC8800. This system can show a user what right or duty he has by answering to the questions on the flow charts of contract law system. This system plays an important role in legal education in the sense that a user can acquire the whole legal system by following the reasoning process. And in 1985, his research group developed another legal expert system, namely, LES-2, a legal reasoning system on contract of sale and law suit game. We will discuss that system later.

3. Characteristics of Legal Knowledge

Legal knowledge has the following characteristics. The *first* is that the conclusion of legal reasoning is constructed as a logical proof. In legal reasoning, justification has a decisive meaning. It means it is necessary to prove that a given decision is a logical conclusion deduced from valid propositions.

The *second* is that legal knowledge is written in natural language. That is, legal world is represented by legal norm sentences written in natural language.

The *third* is that legal knowledge is dynamic and relative. This means that:

(a) The world of events dealt with by law changes with time. Over time, the legal world also changes. Therefore, legal knowledge deals with the world which changes with time.

(b) The legal knowledge itself increases or decreases as time passes. The legal rules, judicial precedents or legal theory is established or has become ineffective.

(c) The content of precedents or theory is dependent on one's view. Therefore, the legal world may vary according to one's position.

(d) The scope of the validity is relative. That is, the validity of the legal norm sentences is limited by temporal, spatial and personal factors.

The *fourth* characteristic of legal knowledge is that it is systematically integrated, which means:

(a) Legal knowledge is composed of legal norm sentences as a unit (4-a).

(b) Legal knowledge has a deep systematic hierarchical structure. Each legal requirement factor is strictly concretized by the legal norm sentence which contains this legal requirement factor as a legal effect (4-b).

(c) Legal concepts, relatively speaking, are well defined semantically and semiotically (4-c).

(d) Meta-knowledge is ordered. For example, legal knowledge has the following meta-knowledge: the validity of legal norm sentences, rule types of legal norm sentences (exemplification or enumeration; positive legal effect or negative legal effect), the sources of legal norm sentences (written law or unwritten law), enforceability of legal norm sentences (enforceable law or

adoptive law). Furthermore, meta-knowledge for reasoning control is ordered. Legal knowledge has meta-legal norm sentence which controls the change of the effect, and the priority of applying legal norm sentences (4-d).

As for the *fourth* characteristic, a further explanation will be given later.

Concerning 4-a: A legal norm sentence makes a form of the structure of a conditional sentence of legal requirement and legal effect. It provides that if a legal requirement is satisfied, a legal effect comes into existence. The logical structure is expressed by the following logical formulae;

```
A:  ∀ X (legal_effect(X) <- legal_requirement(X))
B:  ∀ X (legal_effect(X) <-> legal_requirement(x))
```

Fig.1

Concerning 4-b: In order to make it possible to apply a law to various cases, legal rules are concretized, that is, supported by interpretation of judicial precedents or theory, and abstract legal rules are combined with concrete facts. By adding these legal norm sentences to legal rules, legal norm sentences are logically combined and a legal system can be constructed as follows:

```
0. Legal principle:
     ∀ X(legal_effect(X) <-
        legal_effect1(X) & legal_effect2(X))
1. Legal rule:
     ∀ X(legal_effect1(X) <-
        legal_requirement1(X) & legal_requirement2(X))

1a. Interpretative proposition:
     ∀ X(legal_requirement1(X) <-
        legal_requirement1.1(X) & legal_requirement1.2(X))
```

Fig.2

Concerning 4-d: The unit of legal knowledge is a legal norm sentence. Legal knowledge is composed of a legal requirement and a legal effect (see Fig.2). When a case satisfies the legal requirement of the legal norm sentence, the legal effect comes into existence. In order to determine whether no legal effect comes into existence, when it, on the contrary, does not satisfy the legal requirement(in other words, whether reverse reasoning is possible or not), meta-knowledge is necessary as to whether the legal norm sentence is an enumeration or exemplificating one. In the former case, reverse reasoning is possible, and in the latter case, it is not. The difference between enumeration and exemplification corresponds to the distinction of *equivalence* (Fig.1, B) and *implication* (Fig.1, A) in logical operators. The legal provisions are not always written in that style. But, in legal practice, *equivalence* and *implication* respectively, and essentially represent the same logical relationships as *enumeration* and *exemplification*.

Another important distinction of the rule type is whether the legal effect takes a positive expression or a negative one. This distinction becomes important when a case satisfies two legal norm sentences and produces both a positive legal effect and a negative one, in which case negative one has priority. This distinction controls the priority of conclusion.

Y Furthermore, meta-legal norm sentences control the application of legal norm sentences. In order to resolve a case, it is necessary, first, to define a set of valid legal norm sentences, second, to identify the applicable legal norm sentences among them, and third, to control the priority of application of the legal norm sentences. This reasoning employs meta-legal norm sentences. Meta-legal norm sentences are ordered to apply as follows:

a. An upper law is prior to a lower law.
(principle of upper/lower)
b. An enforceable law is prior to an adoptive law.
(principle of enforceable/adoptive)
c. A new law is prior to an old law.
(principle of new/old)
d. A special law is prior to a general law.
(principle of special/general)
e. A written law is prior to an unwritten law.
(principle of written/unwritten)

Fig.3

Another priority is applicable between these principles.
a. A principle of upper/lower is prior to a principle of special/general.
b. A principle of special/general is prior to a principle of new/old.
c. A principle of special/general is prior to a principle of written/unwritten.

4. The Legal Expert System: LES-2

4.1 The Predecessor of LES-2

In 1982, we began a research project called the *A-Project.* It was a legal reasoning system whose knowledge represented legal norm sentences and the logical relations in the form of logic flow charts written in BASIC and assembler.

This system answers with [(y)es or (n)o] to a user's questions on the logic flow charts on a CRT display, and he can thus reach a legal conclusion. Each display represents a unit of legal requirements and legal effects deduced from one or plural legal norm sentences. However, this system has some problems. It is based on logical analysis, that is, propositional calculus. Therefore, it cannot represent the legal world built of complicated legal concepts, so a computer cannot automatically infer the conclusion.

4.2 Architecture of LES-2 System

The LES-2 employs Prolog-KABA and WING on an NEC PC9801 personal computer. This system was developed by the Legal Expert System Association, whose chairman is Prof. *Yoshino*, one of the collaborators, in cooperation with NEC Corporation. The LES-2 is a prototype of legal expert system on Japanese contract law(*). The system is composed of inference of the substantial law and the lawsuit game. The inference of the substantial law contains the knowledge base of substantial law, the inference engine of the substantial law and explanation module. And the lawsuit game contains the inference engine of procedural law and lawsuit game module. The diagram below shows the major components of the system (Fig.4). In the following sections, these components will be explained.

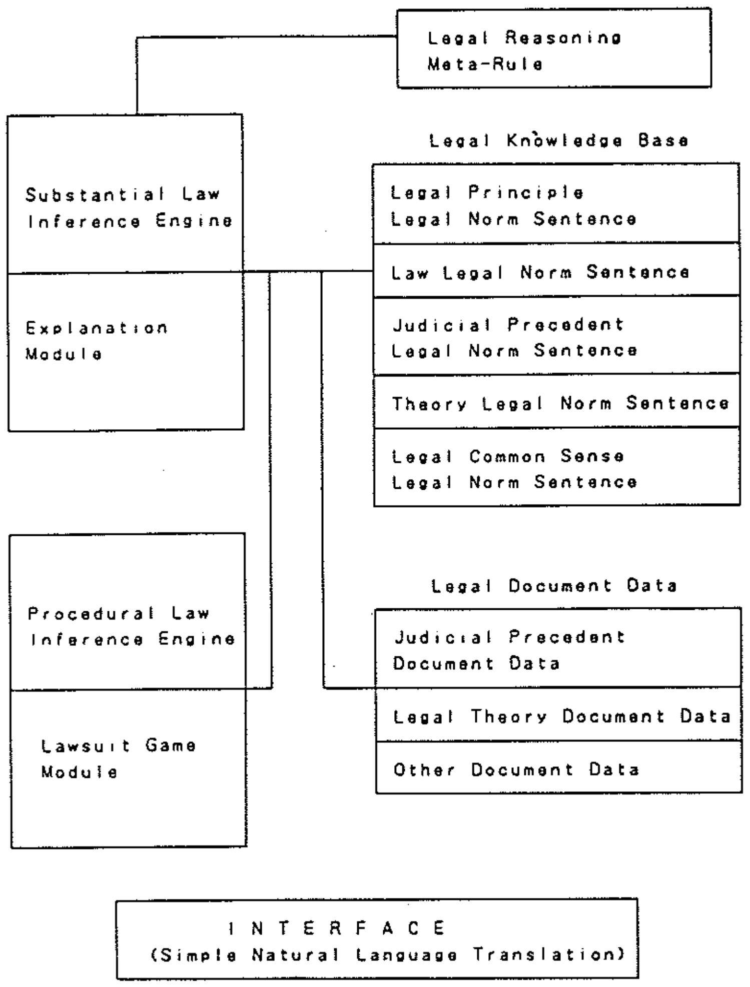
Legal Reasoning Meta-Rule
Substantial Law Inference Engine
Explanation Module
Legal Knowledge Base
Legal Principle Legal Norm Sentence
Law Legal Norm Sentence
Judicial Precedent Legal Norm Sentence
Theory Legal Norm Sentence
Legal Common Sense Legal Norm Sentence
Procedural Law Inference Engine
Lawsuit Game Module
Legal Document Data
Judicial Precedent Document Data
Legal Theory Document Data
Other Document Data
INTERFACE (Simple Natural Language Translation)

Fig. 4

4.3 Substantial Law Inference System

4.3.1 Substantial Law Knowledge Base

In the substantial law knowledge base, legal principle legal norm sentences, law legal norm sentences, judicial precedent legal norm sentences, legal theory legal norm sentences, and legal common sense legal norm sentences are registered. It is mainly here that the characteristics of legal knowledge and the method of the representation according to structure are introduced.

(1) Fundamental Viewpoint of Legal Knowledge Representation

Based on the characteristics of legal knowledge in the previous section, legal knowledge can be formulated under the following fundamental assumptions:
(a) A unit of legal knowledge is a legal norm sentence.
(b) Logical reasoning by natural language is to be performed.
(c) Exact correspondence exists between natural language and formal language for reasoning.
(d) Legal reasoning is a reflection of the legal world.
(e) The structure applicable to increase or decrease knowledge is adopted.

(2) The Representation of Legal Norm Sentences by Compound Predicate Logic Formulae

A legal norm sentence is a unit of legal knowledge, which is formulated in compound predicate logic formulae by Prolog. A compound predicate logic formula expresses the legal effect and legal requirement factors in the whole form as a compound sentence, logical conjunction of the component proposition, by dealing with slot of frame or case grammar as an argument of logic. For example:

legal norm sentence:
it is the following that the contract has become effective at the point of time T2:
at the point of time T1 the effect of the offer has become effective, and

at the point of time T2 the effect of the consent has become effective, and
it is not that at the point of time T2 the effect of the offer has been lost, and
T2 is after T1.

compound predicate logic formulae:

```
has_established(T2,.,legal_act(..,contract(.,.),.)):-
    become_effective(T1,.,declaration_of_intention(..,
    offer(..,contract(.,.)),.)),
    become_effective(T2,.,declaration_of_intention(..,
    consent(..,contract(.,.)),.)),
    not(the_effect_has_lost(T2,.,declaration_of_intention
    (..,offer(..,contract(.,.)),.))),
    after(T2,T1).
```

Fig.5

In order to show characteristics of the compound predicate logic formulae, we compare it with predicate logic formulae based on the literal expression "existence of the legal relation."

compound predicate logic formula by Prolog:

```
exist(ID1,T0,legal_relation(ID2,M1,contract(ID3,M2,M3,M4,
sale),content(ID4,have(ID5,T1,M5,M6,duty(ID6,pay(ID7,T2,
P1,M7,M8,H1,price(ID8,M10,K3,_))))))).
```

predicate logic formulae:

```
VID1 VID2 VID3 VID4 VID5 VID6 VID7 VID8 VT0 VT1 VP1 VM1
VM2 VM3 VM4 VM5 VM6 VM7 VM8 VM10 VK3 VH1(
exist(ID1,T0,ID2)&
legal_relation(ID2,M1,ID3,ID4)&
contract(ID3,M2,M3,M4,sale)&
content(ID4,ID5)&
have(ID5,T1,M5,M6,ID6)&
duty(ID6,ID7)&
pay(ID7,T2,P1,M7,M8,H1,ID8)&
price(ID8,M10,K3,_thing))
```

Fig.6

The representation of legal norm sentences by compound predicate logic formulae provides us with the following advantages.

(a) It can represent the legal effect and each legal requirement factor by one literal as one unit, which compose legal norm sentence. This means that replacing each legal requirement factor on the right side with the head on the left side of Prolog, and it is possible to write directly the rules concretized as the legal effect (see Fig.2). It is effective for writing hierarchical legal knowledge structure combining abstract legal rules and concrete fact.

(b) It can represent knowledge structurally being patterned. By comparing the pattern of the head of rules, it can be used to control the priority of application of *special law > general law*.

(c) It can represent concepts hierarchically, employing the built-in structure of argument of predicate. This enables us to separate general legal norm sentences on the abstract level from concrete legal norm sentences on the individual level, and relate them.

(d) It can formulate the various states of the social relations by writing concepts one by one in detail and composing legal norm sentences.

(e) It becomes possible to formulate legal norm sentences as a unit and in the natural language expressing them; legal reasoning can thus be realized in the system as it is.

(f) It can translate automatically the formulated expressions (compound predicate logic formulae by Prolog) into the representation of natural language, or simple natural language representation, written according to the manual into compound predicate logic formula by Prolog, by allotting the postpositional particles to the argument of the predicate.

(3) The Expression of Meta-Knowledge

Meta-knowledge of legal norm sentences contains legal norm sentences written by compound predicate logic formulae as a rule body. In the knowledge base, the rules are written as follows:

rule(ID,source,theory, validity_range,priority_data, rule_type,application_condition,rule_body).

ID is an identifier of the rule. Source means the name of law from which the rule is derived. Validity_range is an argument set to deal with the rule, which changes spatially, temporally and personally, but the validity_range does not presently work in our system. Theory is the person's name who asserts the rule. This is used to reason the conclusion according to his theory position. Priority_data decides priority of rules as follows:

p(category, intensity, the point of time to create effect)

Category is a category of a legal norm sentence set, to which the given legal norm sentence belongs. It contains the constitution, the civil code, etc., and judges priority using the parameters providing from comparing whether it is *an upper law or a lower law* or *a written law or an unwritten law. Intensity* describes the type of legal norm sentence, to which type of an enforceable law > legal act > an adoptive law it belongs. *The point of time to create* describes the beginning of effect of the legal norm sentence. The inference engine uses these data to control the priority of applying the legal norm sentence. The rule_type is composed of four types, PE, PI, NE, NI, and the sense is explained later. The application_condition describes patterns of compound predicate logic formula. The rule_body is the body of legal norm sentences, and it is provided with disseminating data of the burden of proof in order to be used not only in substantial law inference but also in procedural law inference. Functor *not* has a special meaning of reversing the burden of proof. In Fig.5, fact *not (a declaration of intention to offer the contract has become ineffective)* reverses the burden of proof.

LES-2 employs the concept of a rule system. The rule system is represented in the form of binary relation of rules, and corresponds to priority relation of application and priority relation of conclusion (ground) between rule sets. Furthermore, the rule system is classified as follows: Rule types are classified into enumeration type (E-type) and exemplification type (I-type) on the one hand, and positive conclusion type (P-type) and negative conclusion type (N-type) on the other. By these combinations,

the rule types are classified into positive enumeration types (PE-type), negative enumeration types (NE-type), positive exemplification types (PI-type), and negative exemplification types (NI-type). Consider Fig.7 below:

	enumeration type (E-type)	exemplification type (I-type)
positive rule (P-type)	PE-type	PI-type
negative rule (N-type)	NE-type	NI-type

Fig.7

If the rule body of each rule type is composed of legal requirement (P) and legal effect (Q), the inference engine logically processes them as follows:

PE -type: if and only if P, then Q;
PI -type: if P, then Q;
NE-type: if and only if P, then not Q;
NI -type: if P, then not Q.

To separate the enumeration types from the exemplification types it is assumed that in the former the main rules are E-type and the rest are I-type. In the latter, all the rules are I-type. This distinction is made so that in adding exemplifying rules it is not necessary to know whether existing rules are in enumeration or in exemplification.

4.3.2 Substantial Law Inference Engine

(1) The Function of a Substantial Law Engine

The substantial law inference engine establishes the legal conclusion, which is demonstrated from the legal norm sentences (rules) of the substantial law under the preposition of the true propositions expressing the given case. The substantial law inference engine contains the backward reasoning system of Prolog,to which the priority control between rules has been added.

According to the meaning of the meta-knowledge data represented in the rules, it accumulates the applicable rule set and can determine the applied rules by excluding the other rules using the priority of application. To resolve the goal, the applicability of the rules is determined by the accordance of the goal and the applied conditions. The control of priority of rule application is explained later. Furthermore, as for the other functions, the system has the storage of inference processes for the given questions, explanations, and the output of tracing the inference.

(2) Control of Priority of Rules

Priority relation of rules are of two types, that is, priority of application and priority of conclusion (ground). The priority of application appears when the plural legal rules are applicable to one fact and one of them is given top priority to resolve the goal. For example, this priority can be said of the relation between Art. 97 Sec. 1 and Art. 526 Sec. 1 of the Civil Code of Japan.

Art. 97 Sec. 1 : Declaration of intention becomes effective at the time of its arrival.
Art. 526 Sec. 1: Declaration of intention to accept becomes effective at the time of the dispatch.

In this case, an acceptance is also a declaration of intention, therefore the two rules above concerning the declaration of intention to accept, so that we may have two different conclusions. In the legal world, one of the meta-legal norm sentences (meta-rules) is applicable : *A special law is prior to a general law*. As art. 526 sec. 1, providing declaration of intention to accept, is a special law in relation to art. 97 sec. 1, providing declaration of intention in general, the former is given top priority.

The decision of the binary relation between a special law and a general law depends on comparing the pattern of *application condition*. The two rules above are expressed as follows:

```
Application conditions for Art. 97 Sec. 1:
    has_become_effective(_,_,_,
    declaration_of_intention(_,_,_,_,_))
```

```
Application condition of Art. 526 Sec. 1:
    has_become_effective(_,_,_,
    declaration_of_intention(_,_,_,
    accept(_,_,contract(_,_))
```

Except the *principle of special/general*, the decision of priority relation of application of rules depends on the *priority data* as noted above.

When a rule of E-type has priority, the rule excludes non-prior rules, and when a rule of I-type has priority, the rule is added to the non-prior rules.

Priority of conclusion(ground) is also a kind of rule priority. It involves a binary relation between the rules with positive legaleffect and the rules with negative legal effect. It plays a role in deciding which conclusion of rules is prior when the proof of both rules is successful and the conclusions are inconsistent. In legal practice, an exceptional rule is prior to a principle rule, and a rule with negative legal effect is prior to a rule with positive legal effect. For example, this priority can be said to exist between Art. 93 of Civil Code and the proviso.

Art. 93: A declaration of intention shall not be invalidated by the fact that the declarant has made it knowing such declaration not to be his real intention;
Proviso: However, such declaration of intention shall be null and void, if the other party was aware, or should have been aware, of the real intention of the declarant.

When a fact satisfies the requirement of the proviso of Art. 93, it also satisfies the main text. In this case, it is impossible to apply either of them previously by comparing application conditions. When the goal is inconsistent with the resolved conclusion, the proviso of negative conclusion is applied.

Priority of rules is controlled by the inference engine in the following way. The inference engine, first, extracts the rules applicable to a case and accumulates the rule set. If the rule set contains a rule of the positive enumeration type, the engine extracts the rule of top priority according to a meta-rule controlling priority relation of application and accumulates the rule set again.

If this rule set contains the rule of negative enumeration type, the engine accumulates again the rule set by excluding the rule of negative enumeration type below that level.

The rule set accumulated in this way is classified into a positive rule set and a negative rule set. Furthermore, the rule set to prove the goal is classified into a positive rule set (Rp) and a negative rule set (Rn), and according to the rule type above said, the truth or falsity of goal (G) is determined as follows:

a. If Rp is not a null set and Rn is a null set,
   G is true.
b. When both Rp and Rn are null sets, G is false,
   if Rp involves E-type rules, and G is not defined,
   as the rule is exemplifying if otherwise.
c. If Rp is a null set and Rn is not a null set,
   G is false.
d. When neither Rp or Rn are empty sets,
   G is true if a positive rule is prior, and
   G is false if a negative rule is prior.
   In the other case the knowledge structure is not
   well-defined.

Meta-rules to control priority of rules are written not in the substantial law knowledge base but directly in the inference engine.

4.3.3 Substantial Law Explanation Module

Substantial law explanation module shows the explanation of inference process using the data structure representing the inference process given from the inference engine. Therefore, a user can search for the node of proof tree of inference. But, as to the rule of N-type, a user can research for the proof tree at the time of failure by not_goal.

4.4 Procedural Law Inference System

4.4.1 Procedural Law Inference Engine

Procedural law inference engine aims to infer a conclusion of procedural law, putting in the argument or evidence of the parties and giving truth value to the procedural law propositions. A higher order rule is used for expressing the processed and determined by an inference engine according to an expression form of substantial law rules. Moreover, the process structure of a suit and flow are also built into the inference engine.

4.4.2 Lawsuit Game Module

The lawsuit game module can simulate a legal case. Using a procedural law inference engine, it infers the condition of suit action (here, assertion and plea) of litigants (plaintiff, and defendant) and the weight of evidence in a civil action. Furthermore, it models the interlocutory judgment according to the proceeding of the lawsuit and finally the conclusion of the definitive judgment.

4.5 Interface

When the interface of a substantial reasoning system is displayed, the display will have the commands, *establishment of a case*, *representation of a case*, *establishment of the goal to be resolved*, *performance of reasoning*, *explanation.* It employs the functions of *question and answer*, and *reference of the related legal document data.* The interface of law suit game realizes the function to input *object of claim*, and *oral pleadings*.

Both interfaces are provided with the function of simple natural language translation for input and output in Japanese. This method can automatically be translated into compound predicate logic formulae natural language which is put in with parentheses and space, according to the manual into Prolog sentences. It can, on the other hand, translate the compound predicate logic formulae into natural language as well.

5. Conclusion

LES-2 was not intended to produce a practical reasoning system, but was intended as a pilot system in order that we might investigate various possibilities and problems of developing a legal expert system for practical use in the future. Through our research, we have proven the possibility of developing a practical system.

In building this system, we were able to clarify the structure of legal knowledge by way of compound term and analyzing the relation of words used in law. To resolve the priority order among different laws and provisions (articles), we employed the priority control by meta-knowledge and meta-rules.

But we think the priority control by meta-rules should be resolved as logical proof and thus we must make a legal argument machine representing the meta-rules as rules.

It is now important that we should carry forward a full scale development of legal expert system. For the study of law, legal education, and legal practice need such a practical AI system in performing legal services. At the same time, we should aim to realize a legal artificial intelligence gifted with a legal mind, so that a computer with *human* AI may be developed. The legal mind is one of the most intellectual abilities of human beings, which, it has been demonstrated, is ready to be analyzed and systematized by a computer.

Acknowledgments

(*) These eight researchers joined to develop the LES-2:
H. Yoshino (Professor of Law, Meiji Gakuin University, Tokyo), *S. Kagayama* (Assistant Professor of Law, University of Osaka, Osaka), *S. Ohta* (Associate Professor of Law, University of Nagoya, Nagoya), *M. Kitahara* (Lecturer in Law, Meji Gakuin University, Tokyo), *H. Kondoh* (C&C Systems Research Laboratories, NEC Corporation, Kawasaki), *M. Nakakawaji* (Kobe NEC Software Inc., Kobe), *K. Ishimaru* (NEC Corporation, Tokyo), *T. Takao* (NEC Corporation, Tokyo).

References

H. Yoshino, Die Logische Struktur der Argumentation bei der Juristischen Entscheidung, in: A. Aarnio u.a. (Hrsg.), Methodologie und Erkenntnistheorie der juristischen Argumentation, Rechtstheorie Beiheft 2 (1980), S. 233 ff.

H. Yoshino, An Application of Computers to Reasoning in the Judicial Process. Law and Computers No. 3 (1984), p. 77 ff.

H. Yoshino, A Possibility of Legal Expert System. ICOT Commission Research Report '86 (1986).

H. Yoshino, Legal Expert System as a Legal Reasoning. 86-IS-11 Proceedings of Japan Society for Software Science and Technology Vol. 86, No. 43 (1986), p. 1-11.

H. Yoshino (ed.), Foundation for Legal Expert Systems, Legal Theory 1, GYOSEI, Tokyo 1986.

H. Yoshino/S. Kagayama/S. Ohta/M. Kitahara/H. Kondoh/M. Nakakawaji/K. Ishimaru/S. Takao, Legal Expert System LES-2, in: Proceedings of the Logic Programming Conference '86, p. 67 ff.

J. Ikeda, Inheritance Taxes Law Expert System, in: H. Yoshino-(ed.), Foundation for Legal Expert Systems, Legal Theory 1, GYOSEI 1986, p. 59 ff.

K. Nitta, Patent Law Expert System, in: H. Yoshino (ed.), Foundation for Legal Expert Systems, Legal Theory 1, GYOSEI 1986, p. 80 ff.

M. Ikeda & H. Hozumi, Copyright Law Expert System, in: H. Yoshino (ed.), Foundation for Legal Expert Systems, Legal Theory 1, GYOSEI 1986, p. 71 ff.