Invited Paper

Representation of Legal Knowledge

KATSUMI NITTA*, MAKOTO HARAGUCHI** and SEIICHIRO SAKURAI**

To develop a knowledge base for social domains, one should understand the essential features of domain knowledge. In this paper, we describe the reasoning and knowledge of lawyers. To solve legal problems, lawyers use available knowledge and draw conclusions by complex reasoning. As it is difficult to simulate lawyers', reasoning steps accurately by means of a simple rule-based system, we propose two reasoning mechanisms. One is a rule-based system that can generate interpretations, and other is a legal reasoning mechanism with rule-based reasoning and case-based reasoning. We give examples of the solution of actual legal problems with these models.

1. Introduction

In social domains such as law, trade, economy and education, human experts use various kinds of knowledge to solve problems. To develop a knowledge base for a social domain, we must understand its essential features. In this paper, we select legal knowledge as a typical example of social knowledge and analyze its features.

The main difference between legal and physical knowledge is that legal knowledge is artificial. Since ancient times, people have devised rules to solve problems. Such rules have been systematized as legal systems. Therefore, legal knowledge and its inference mechanisms seem to be easier to analyze than physical knowledge. However, legal rules are insufficient to solve real problems automatically, and real problems contain very complex situations in which some facts may be missing and some assumptions may be subjective. Nevertheless, lawyers can draw some consequences in lawsuits because they use a variety of legal knowledge in addition to legal rules.

Though legal reasoning has been long studied as an application of artificial intelligence technologies, there is still no definite model. In this paper, instead of reviewing various legal systems, we will introduce our systems and show our legal reasoning model. By using some examples, we will explain why legal reasoning systems are hard to develop.

In Chapter 2, we will describe the legal activities of lawyers and the knowledge and reasoning they use. We will explain why legal reasoning is not modeled as a simple rule-based system. Chapter 3 introduces a rule-based system called LES that tries to realize an interpretation mechanism. In Chapter 4, a case-based system called HELIC-II is introduced. HELIC-II tries to make up for the rule-based reasoning by using case-based reasoning.

2. Legal Knowledge and Legal Reasoning

2.1 Lawyers' Reasoning Processes

(1) Lawyers' Activities and Interpretation of Law

Lawyers' activities cover a wide range of legal problems. For example, they word contracts, make arguments in lawsuits, negotiate amounts of damage, and help with taxes.

When wording contracts, they try to protect the client from any foreseeable problems. In this case, they use not only their knowledge of contract law but also their knowledge based on experiences related to contract law. For example, they may remember previous difficulties caused by ambiguous sentences in a contract, and they may have a set of typical sentences from past contracts to use.

When they formulate arguments for use in lawsuits, the goal is given by the client. Lawyers try to make rational explanations that lead to the goal and persuade the jury. In this case, they refer to old cases and theories concerning the interpretation of legal rules, and try to find evidence that supports their explanations.

The actions of lawyers, however, are not limited to the court, and they use a variety of information in addition to statutory law. Here, we focus on knowledge related to the application of law, because it is the most relevant to the law. In this field of knowledge, interpretation of law is the most common area in lawsuits.

(2) Legal Systems

Different countries have different legal systems. Japan and many countries in western Europe employ

*Institute for New Generation Computer Technology.
**Tokyo Institute of Technology, Tokyo, Japan.

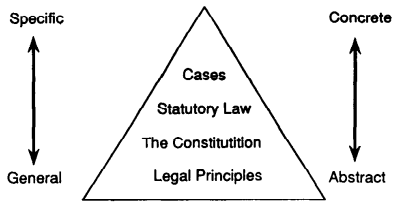

Fig. 1 Legal System.

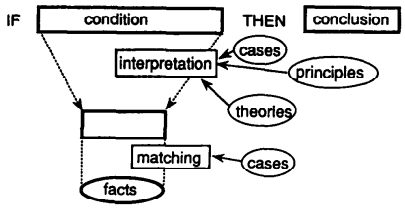

Fig. 2 Interpretation of Law.

rule-based systems of law. On the other hand, the United States, Canada, and Britain employ common law systems. In the former system, statutory law is an important knowledge source, and new cases are solved deductively. In the latter system, precedents are more important, and new cases are solved inductively. Though there are different legal systems, the inference processes of lawyers have many things in common. In this chapter, we will concern ourselves mainly with explaining legal knowledge in rule-based systems of law; most of this discussion will also be applicable to common law systems.

A legal system consists of various legal knowledge sources, such as legal principles, statutory laws, and precedents (old cases), which are organized into a complete legal system. For example, to realize legal principles, constitutions are created, and statutory laws are based on these constitution. The judges in court draw conclusions based on statutory laws (Fig. 1). The knowledge in each layer is given in the form of rules. However, there are gaps between layers, because predicates in the lower layer of Fig. 1 are more abstract than those in the higher layer, and there are no rules for combining different layers.

In statutory law, there are several systems such as civil law and criminal law. They are constructed as different systems. For example, the civil law system consists of civil code and commercial law. As commercial law is a field of civil code, its rules normally have priority over those of civil code.

(3) Statutory Law and Interpretation

A statutory law consists of legal rules, most of which are given in the form of IF-THEN rules. If the given facts of a new case use exactly the same predicates as legal rules, and if they satisfy the conditions of a legal rule, we can draw legal conclusions by applying legal rules deductively. However, in actual cases, it is rare for legal consequences to be obtained by deductive reasoning alone. The reasons for this are as follows:

- Legal rules contain many legal predicates. Legal predicates are often ambiguous and their strict meanings are not fixed until the rules are applied to actual facts. For example, the meanings of "public welfare," "due reason,", "good morals," and "in good faith" cannot be defined explicitly beforehand. Their meanings depend on social conditions such as the economic policy of the government, the emotional disposition of the people, and so on.
- Since legal rules are given as sentences in natural language, they contain grammatical ambiguities.
- More than one rule may be applicable to a new case, and inconsistent conclusions may be drawn as a result.
- Some rules may be obsolete when applied to new cases. For example, when the copyright law was litigated, there were no computers. Therefore, rules in the copyright law may be inappropriate for protecting computer programs.

Though legal rules involve the above problems, lawyers can draw conclusions by following legal reasoning. To apply legal rules to actual facts, lawyers make the meanings of legal predicates clear before matching legal rules and facts. This task is called *interpretation* of a law. To do this, lawyers may have to consider not only each rule but the role of each rule in the whole rule system. If two rules are inconsistent, lawyers may extend or reduce the meaning of a legal predicate to adjust the effect of each rule. In extreme cases, they may have to make a temporary rule to solve the given case.

The interpretation of law involves operations to determine the meaning of a legal rule dynamically by using techniques such as literal interpretation, completion, generalization, and analogical interpretation (Fig. 2).

The interpretation of law is realized in two stages—the generating stage and the selecting stage. In the generating stage, lawyers generate various interpretations from different viewpoints. In the selecting stage, lawyers select the most rational interpretation. To select the proper interpretation, they must evaluate each interpretation from various viewpoints, such as social needs and the general principles of law. However, there is no definite way to select the proper interpretation.

(4) Knowledge for Interpretation

We have explained the interpretation of law briefly. In each stage of law interpretation, lawyers use the following knowledge. Since this knowledge has no legal effect constraining a judge's decision, the use of such knowledge is left to the judge's discretion.

- Legal Principles:
  Legal principles are primitive doctrines that declare what the law should protect. For example, "protecting public welfare" and "guaranteeing fundamental human rights" are typical principles.

Since some principles conflict with each other, there are other principles to balance them. For example, the "principle of fairness" balances the "principle of protecting the rights of minorities" and the "principle of protecting the stability of trading."

Some appear as they are in the constitution, but most appear as more concrete rules in statutory law. Since original legal principles are rules that are too abstract to apply to actual cases, they are usually used to evaluate the appropriateness of an interpretation of legal rules.

- Old Cases:

  Judicial precedents consist of the following information:

  —Text: (final conclusion)
  —Facts and points at issue: (circumstances of the case, and arguments of both parties)
  —Reasons: (opinions of the judges)
  —Parties and their legal representatives:
  —The court:

  When lawyers wish to argue a point or to support their case, they search for similar precedents. As the arguments and conclusions in similar cases tend to be similar, precedents are important knowledge sources for predicting the arguments and final conclusion.
- Interpretation Rules:

  Interpretation rules are rules extracted from old cases and generalized to fit other cases. As there is much redundant information in old cases, it is not easy to extract reliable rules that affect the final conclusion and are applicable to cases. These rules define more concretely the meanings of abstract predicates in statutory law.
- Theories for Interpretation:

  Many cases involve similar situations. Scholars of law schools have tried to make theories that categorize cases into several patterns and give unified explanations for them. These theories are used to guide new interpretations.
- Aims and Roles of Statutory Laws:

  When two rules are applicable to a new case and they conflict, the aim or role of each rule is often referred to and used to interpret the rule. Though the aim does not appear explicitly in the law, it is found in legal textbooks.
- Dictionary of Legal Concepts:

  For each concept appearing in the law, related concepts such as the upper concept, the lower concept, antonyms, and synonyms are often referred to. In particular, interpretation by completion or analogical reasoning requires this knowledge.
- Social Customs, Common Sense, and Industrial Policies:

  Rules related to social customs, common sense, and industrial policies affect the interpretation of legal rules.

Most of this knowledge is given as rules. However, these rules do not constitute a consistent overall rule system. Therefore, even if we construct a rule base using statutes and this knowledge, it is still insufficient to solve legal problems.

2.2 Reasoning Model and Knowledge Representation

Since law is a field in which the rule set is insufficient, we need a reasoning model to supplement simple rule-based reasoning, which merely performs forward chaining. There are two ways to supplement simple rule-based reasoning. One way is to expand rule-based reasoning so that it can handle analogical reasoning. This mechanism corresponds to analogical interpretation of law. The legal reasoning system LES presented in Section 3 employs this mechanism.

Another way to model the interpretation step is to use *case-based reasoning*. This mechanism corresponds to the fact that lawyers often refer to the results of similar old cases and use them to construct arguments. The legal reasoning system HELIC-II, presented in Section 4, employs this approach. Since case-based reasoning complements rule-based reasoning, legal reasoning is modeled as a combination of rule-based reasoning and case-based reasoning.

The following are general problems in constructing a rule base and a case base:

- A rule base:

  Statutes are published and can be easily accessed. Since each legal rule is given as a natural sentence, there may be ambiguities in the rule, or two rules may be inconsistent with each other, if taken literally. Therefore, we may have to reform each rule to resolve inconsistencies, or we may have to use a higher-order reasoning mechanism such as nonmonotonic reasoning.

  In the rule base, a variety of information appears, such as the temporal relation between events, modalities such as "may," "shall," and "must not," the notion of "believe" and "know," and the concept of causality between events and states. Sometimes, we need to discriminate logical negation and negation as failure in the negation of legal rules. To handle these problems, a powerful knowledge representation language and a powerful inference mechanism are needed.
- A case base:

  Since old cases are published, they can be accessed through databases or books. Old cases contain a variety of information, such as circumstances (facts), the arguments of both parties, the judge's opinion, and final conclusions. However, since these are raw data in the form of sentences in natural language, we must change them into forms that computers can manipulate, if we are to use this in-

formation in an automated reasoning system. Usually, lawyers use only some of the information from a case. They may use only the final conclusion, the opinion of the judges, or the arguments of both parties.

Since only a few case-based reasoning systems have so far been developed, representation of cases, indexing, retrieval, adaptation, and repairing are still open problems. Generally speaking, if the cases are represented in detail, the evaluation of similarities becomes difficult. Therefore, deciding on the proper primitives for representing cases is important.

3. Rule-Based System for Statutes

As we showed in our general survey in Section 2, we need various types of knowledge to achieve our legal reasoning system. Knowledge for interpreting legal rules is considered especially important. According to standard textbooks on law [7, 1], the interpretations are usually classified into four major categories: literal interpretation, completion, generalization, and analogical interpretation (or analogy). Literal interpretation interprets legal rules as literally as possible. Completion is equivalent to assuming the "only-if" parts of the rules. These two interpretations thus restrict the applicabiliy of legal rules. On the other hand, both generalization and analogy extend their applicabilities. This section first describes the last two interpretations, presents an actual case in which the judges applied analogy, analyzes knowledge to justify the analogy, and finally shows how the knowledge is utilized in a symbolic reasoning system.

3.1 Generalization and Analogy

Generalization can be understood as an act of replacing some requirement A in legal rules with a more general requirement B. To determine whether B is more general than A, we need a legal domain theory consisting of various types of knowledge, as mentioned in Section 2.1. A conceptual hierarchy involved in a dictionary of legal concepts is especiatly useful for this purpose. We will present a real example in the following sections.

Under such a domain theory, the statement that B is more general than A is expressed as $A \rightarrow B$, where $\rightarrow$ denotes a logical implication. Our generalization can then be illustrated as in Fig. 3, where $A \rightarrow X$ denotes a legal rule with X and A as its effect and requirement, respectively.

$$\frac{A \rightarrow X \quad A \rightarrow B}{B \rightarrow X}$$

Fig. 3 Generalization Schema.

$$\frac{A \rightarrow X \quad A \sim_{\alpha} B}{B \rightarrow X}$$

Fig. 4 Analogy Schema.

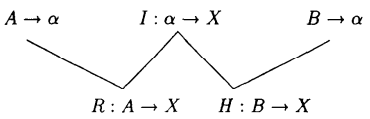

Fig. 5 Analogy Diagram.

On the other hand, analogy is said to be an act of replacing a requirement A with another requirement B, provided that these two requirements are similar with respect to some important legal aspects.

Let us examine Fig. 4 for a while. Given a current case subsumed by B, suppose that a lawyer has a target goal in mind, which is an intended legal conclusion not derived from the case under the present domain theory. In other words, the domain theory is too weak to reach the desired conclusion. Then, the problem is to make the hypothesis $B \rightarrow X$ in Fig. 4 in order to obtain X for the present case. The logic used to justify such an analogy can be stated as follows [8]: The legal rule $B \rightarrow X$ is considered valid because $\alpha \rightarrow X$ is valid and because $A \rightarrow \alpha$. Moreover, since $B \rightarrow \alpha$, we can conclude that the same effect X is derived from B by assuming the existence of a common α. It should be noted here that the intermediate hypothesis $\alpha \rightarrow X$ represents the heart of the law in a sense. The situation is more precisely illustrated by the diagram shown in Fig. 5.

Figure 5 consists of two sub-diagrams with the same V shape. The left V is to obtain an intermediate hypothesis I: $X \leftarrow \alpha$ by a generalization. The other V means a deduction to get the final hypothesis H, which is the result of analogical interpretation.

As we have just observed, the primitive operators needed to realize analogical interpretation of legal rules are generalization and deduction. Needless to say, deductive engines are necessary for almost all legal reasoning systems. For instance, LES [6] is a logical system for performing deductive reasoning under a set of legal rules encoded as Horn clauses. The system is, therefore, implemented in resolution-based deductive engines, especially in Prolog. On the other hand, generalization has been studied in the field of machine learning, and various computational operators for generalization have been proposed by many researchers. Since the reasoning system we expect should be basically deductive to execute various types of knowledge, we choose an absorption operator [4] to realize our generalization. This is because absorption is a kind of inverse resolution and is therefore easily implemented in resolution-based deductive engines. A more detailed description of absorption, deduction, and analogy can be found in [2].

We are now ready to discuss a way of representing legal rules and other related knowledge. In Subsections 3.2 and 3.3, we first introduce the knowledge representation language of LES, and try to represent Article 94 of the Japanese civil code, for which an analogy is applied. The representations we present here are almost the same as those given by Yoshino. et al. [5]. However, they have been revised to capture the notions of falsity and transfer of rights more precisely.

3.2 Knowledge Representation in LES

LES is a logical system that performs deductive reasoning under a set of Horn clauses. This means that both legal rules and other rules for interpreting the requirements of rules are encoded by Horn clauses. LES, in fact, has two levels of representation languages. The first describes formalized legal rules so that their intended meaning is as close as possible to the real meaning of the articles. Each predicate, called a compound predicate, has the form

⟨*predicate name*⟩ (⟨*relation identifier*⟩, ⟨*list of slots*⟩),

where *slot* is a pair of case slot name and its value. The value of a slot may be a variable or a compound predicate. Thus the language allows us to use case representation just like case grammar. Moreover, each predicate instance has an identifier, a *relation identifier*, to refer the instances of other predicates. The identifiers correspond exactly to pointers for connecting frames in frame-based systems.

Each legal rule is written as a definite clause in which atomic formulas are compound predicates, $X, A_1, \ldots, A_n$:

$$X \leftarrow A_1, \ldots, A_n$$

The second level of representation language is the language defining Prolog clauses. First-level clauses are automatically translated into standard definite clauses in Prolog.

For instance, Clause 2, Article 94 of the Japanese Civil Code, which is one of the most frequently applied rules in legal analogy, is shown in Fig. 6. Clause 1 means that a declaration is invalid if the parties agree that the declared intention was different from the real intention. Clause 2 means that the invalidity of a declaration of intention cannot be set up against an act of good faith, where the person involved did not know the fact that the declaration was different from the real intention. The purpose of Clause 2 is to protect individuals who may have believed a declaration of intention.

The legal rule, which is translated from Clause 2, Article 94, in Fig. 6, is shown in Fig. 7. The rule was not directly translated from a legal sentence but compiled with legal knowledge. In other words, the rule in Fig. 7 corresponds to a rule interpreted by a lawyer.

In Fig. 7, symbols, which are preceded by capital letters, represent logical variables. The predicate *contract*

Article 94 (negotiated false declaration of intention):
Clause 1 Negotiated false declaration of intention is invalid.
Clause 2 The invalidity of declaration of intention, which comes under the previous clause, cannot be set up against an act of good faith, where the person involved is not aware of the falsity of the declaration.

Fig. 6 Article 94 of the Japanese Civil Code.

contract(*Ctrct*1, [*parties*: [*X*, *Y*]]),
falsity(_, [*obj*: *Ctrct*1]),
contract(*Ctrct*2, [*parties*: [*Y*, *Z*]]),
good_faith(_, [*agt*: *Z*, *obj*: *falsity*(_, [*obj*: *Ctrct*1])])
→*cannot_set_up*(_, [*agt*: *X*, *god*: *Z*, *obj*: *Ctrct*2])

Fig. 7 Example of a Legal Rule (Clause 2, Article 94).

Judgment: By analogical application of Clause 2, Article 94 of Japanese Civil Code, "A" cannot set up "C," who acted in good faith and did not know the real intention of "A." Nevertheless, "B" does not get ownership.

Fig. 8 Example of Analogical Application.

represents some contract. The predicate *falsity* means that the declaration of intention is different from the real intention in its object's value, *Cntrct*1. The predicate *good_faith* is defined as follows:

good_faith(*Id*, *Contnts*)←*restricted_not*(*Contnts*)

The predicate, *restricted_not*, in the above definition is defined in the next subsection. The predicate *cannot_set_up* means that *X* cannot claim the invalidity of the contract, *Cntrct*2, because of the invalidity of the contract, *Cntrct*1. Note that this rule is applied in spite of the legal invalidity of its requirement part, *contract*(*Cntrct*1, [*parties*: [*X*, *Y*]]).

In legal reasoning, it is importants to represent the facts that constitute a case. Before describing the representation of facts, a leading case of the analogical application of Clause 2, Article 94 is shown in Fig. 8. In the case, since "B" sold the house without having legal ownership, "A," who was the real owner, claimed his ownership right. However, the judge decided by legal analogy that "*C*" had the real ownership right, in spite of the invalidity of the sale between "*A*" and "*B*".

Case of a petition against registration of passage of a house's title ((O) No. 107–1951, judgment of the second petty bench, Aug. 20th, 1951))

Case: After "A" bought a house, which "O" owned, from "O", he approved the registration of passage of title from "O" to "B" without showing his real intention in seeking passage. After registering the passage, "B" sold the house to "C", who bought the house in good faith without knowing the real intention of "A". Ownership was registered as belonging to "C." "A" claimed that "C" must cancel the passage of title, because the ownership of the house should belong to "A" and neither "B" or "C" had

sale_of_immovables(*id*1, [*parties*: [p_o, p_a], *obj*: imm_X]).
ownership(*id*2, [*agt*: p_a, *obj*: imm_X]).
reg_of_ptitle(*id*3, [*parties*: [p_o, p_b], *obj*: imm_X]).
reg(*id*4, [*agt*: p_b, *obj*: imm_X]).
approval(*id*5, [*agt*: p_a, *obj*: *id*6]).
sale_of_immovables(*id*6, [*parties*: [p_b, p_c], *obj*: imm_X]).
reg_of_ptitle(*id*7, [*parties*: [p_b, p_c], *obj*: imm_X]).
know(*id*8, [*agt*: p_c, *obj*: *id*3]).
know(*id*9, [*agt*: p_c, *obj*: *id*4]).
know(*id*10, [*agt*: p_c, *obj*: *id*6]).
know(*id*11, [*agt*: p_c, *obj*: *id*7]).

Fig. 9 Facts of the Case in Fig. 8.

ownership of the house.

Legal cases are formalized by a set of approved facts, and an approved fact is also represented by a compound predicate. An exampleis shown in Fig. 9. In Fig. 9, p_a, p_b, p_c, and p_o represent the agents in the case. The first argument of each predicate is an identifier for reference and the second argument is a list of pairs of a case and a value. The name, "agt:", represents an agent case and "obj:" represents an object case. The predicate *reg_of_ptitle* means "registration of passage of title" and "reg" means "registration." The predicate "know" represents the fact that its agent knows the object. Details on the use of "know" are described in the next subsection.

Other knowledge is needed to realize legal reasoning, and is also represented by rules and facts.

3.3 Analysis of Analogical Application and Knowledge Needed to Realize Legal Analogy

Let us consider the case in Fig. 8 again. Obviously, if the person acting in good faith, "O," did not appear, the ownership of "A" would be approved. Then, what inference can be made in this case? What knowledge is needed to realize such a legal analogy?

To clarify why the rule cannot be applied to the case in Fig. 8, let us consider the difference between Clause 2, Article 94 and the case. Their relation is shown in Fig. 10. If O in Fig. 10 is omitted, we can see a trinary relation, and one-to-one partial correspondence, [*A-X*, *B-Y*, *C-Z*, . . .}, can be found. However, although the rule has a relation between X and Y as a legal requirement, the case has no relation between A and B. What relation corresponds to the contract between X and Y or the falsity of the contract? In such a case, the judge can transform the case with his common-sense or legal common-sense knowledge. Therefore, legal analogy requires some knowledge in order to transform a case description so that the correspondence between a case and a rule can be found easily.

The transformed case is shown in Fig. 11. In Fig. 11, two virtual relations of passage of title are introduced. In other words, we can assume that there was a passage of title from O to A and a passage of title from A to B, although they were not registered formally.

If the case is transformed as shown in Fig. 11, can we apply Clause 2, Article 94? Obviously, a relation, which

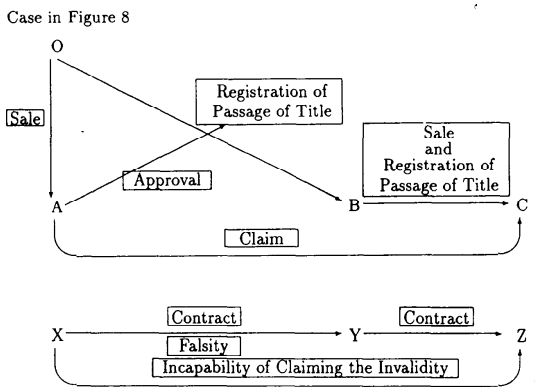

Fig. 10 Relation between the case in Fig. 8 and Clause 2, Article 94.

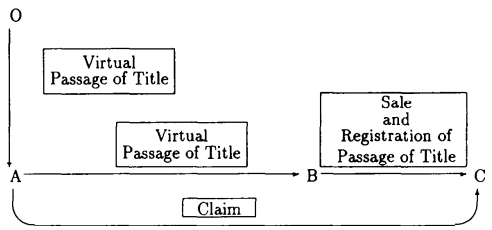

Fig. 11 Transformation of the Case in Fig. 8.

contract(*Ctract*, _)
repr_of_ctrct(_, [*obj*: *Ctrct*, *Atr*: *ReprCts*]),
soa_of_ctrct(_, [*obj*: *Ctrct*, *Atr*: *RealCts*])),
ReprCts ≠ *RealCts* → *falsity*(_, [*obj*: *Ctrct*])

Fig. 12 A Rule of a Theory of Interpretation(False Declaration).

corresponds to the falsity in Clause 2, Article 94, is missing. In legal reasoning, a missing relation prevents the application of a rule; the relation must then be fulfilled by some knowledge before application of the rule. We can assume that theories of interpretation will form such knowledge. Rules of the theory of interpretation are also written as compound predicate rules. Figure 12 shows an example of such a rule. The rule is paraphrased as:

> If the content of the representation of a contract is different from that of the state of affairs of the contract, then it is concluded that the contract has falsity.

In Fig. 12, a slot variable, *Atr*, is used to represent the variable content. The predicate, *repr_of_ctrct*, represents the "representation of the contract" and the predicate *soa_of_ctrct*, represents the "state of affairs of the contract."

If the rule in Fig. 12 is assumed,a relation may be introduced. However, to introduce the relation, falsity, more knowledge must be assumed, since the rule is not

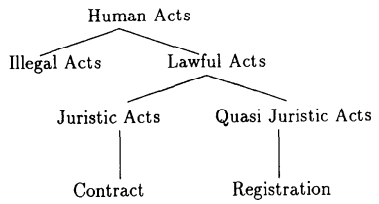

Fig. 13 Example of Classification Knowledge.

Textbook knowledge

$contract(Id, Cts) \rightarrow juristic\_act(Id, Cts).$
$juristic\_act(Id, Cts) \rightarrow lawful\_act(Id, Cts).$
$reg\_of\_ptitle(Id, Cts) \rightarrow registration(Id, Cts).$
$registration(Id, Cts) \rightarrow quasi\_juristic\_act(Id, Cts).$
$quasi\_juristic\_act(Id, Cts) \rightarrow lawful\_act(Id, Cts).$

Common Sense Knowledge

$reg\_of\_ptitle(Regp, [parties : [\_, Agt], obj : Obj])$
$record\_ownership(\_, [agt : Agt, obj : Regp]),$
$\quad \rightarrow repr\_of\_reg(\_, [obj : Regp, agt : Agt]).$
$reg\_of\_ptitle(Regp, [\_, obj : Obj]),$
$ownership(\_, [agt : Agt, obj : Obj]),$
$\quad \rightarrow soa\_of\_reg(\_, [obj : Regp, agt : Agt]).$

$repr\_of\_ctrct(\_, Cnts) \rightarrow repr(\_, Cnts).$
$soa\_of\_ctrct(\_, Cnts) \rightarrow soa(\_, Cnts).$
$repr\_of\_reg(\_, Cnts) \rightarrow repr(\_, Cnts).$
$soa\_of\_reg(\_, Cnts) \rightarrow soa(\_, Cnts).$

Fig. 14 Description of Classification Knowledge.

about the declaration but about the contract. In order to fill the gap between the declaration and the contract, classification knowledge may be needed. Figure 13 shows an example of classification knowledge.

The knowledge in Fig. 13 is text-book knowledge. In order to fill the gap, another type of classification knowledge is needed. Figure 14 shows an example of such knowledge. In the figure, the predicate "*record_ownership*" represents recorded ownership, which may be different from real ownership.

Classification knowledge plays another important role in legal analogy. Even if virtual relations are correctly introduced with theories of interpretation, the number of possible pairings between the case and the rule is huge. To reduce the number and obtain an adequate pairing, classification knowledge can be used. For example, we can obtain a pairing between the contract and the passage of title, since the upper class of both concepts is a lawful act.

3.4 Negative Predicates

The predicate *good_faith* needs special treatment, because good faith means that members of the involved party are not aware of the falsity of a negotiated contract. We call such a predicate negative, and assume that the predicate should be declared a priori. No other rule except negation-as-failure is applied to negative predicates. Here it should be noted that only facts known by the members of a party can be available in invoking the negation-as-failure rule. A fact f known by an agent p is explicitly designated as a compound predicate instance:

$$know(\_, [agt: p, obj: f]).$$

Moreover, negative predicates are assumed to be evaluated only after their arguments, except the identifier, have been completely instantiated. This is because there are many more negative facts than positive facts. Thus, early evaluation of negative predicates generally increases the size of the search space. For this reason, we use the strategy to delay the evaluation of negative predicates.

declaration of negative predicate
negative(good_faith)

Once the predicate has been thus declared, the system automatically expands the definitional rule:

$$good\_faith(Id, [agt: Agt, obj: Fact])$$
$$\leftarrow restricted\_not(Agt, Fact)$$

Now, the meaning of *restricted_not(Agt, Fact)* is clear. That is, it succeeds only if the goal *Fact*, which should be a compound predicate instance or its identifier, finitely fails under a set of facts F such that *know*(_, [*agt*: *Agt*, *obj*: F] is presented. When *Fact* or F are identifiers, we must find the compound predicate instances that they are referring to. For this purpose, we need to record a reference relation between them. However, it suffices to use a method similar to the blackboard mechanism, which can be easily implemented.

3.5 Utilization of Legal Knowledge by Generalization and Deduction

We are now ready to shw how to utilize legal knowledge and how to carry out legal reasoning mechanically.

First, suppose that we are seeking a legal explanation to protect the rights of a person acting in good faith, p_c, who appeared in the case presented in Subsection 3.2. Since the plaintiff p_a claimed that p_c should cancel the passage of title, it suffices to show that p_a cannot set up p_c with respect to the passage of title. Hence, we show the following goal:

$$cannot\_set\_up(Id, [agt: p_a, goa: p_c, obj: id7]),$$

where $id7$ is the identifier of the registration of passage of title from B to C. Our system first retrieves a rule whose head predicate is *cannot_set_up*. From Clause 2, Article 94, we obtain the next goal-list:

$$contract(Ctrct1, [parties: [p_a, Y]]), \quad (1)$$
$$falsity(Falsity, [obj: Ctrct1]), \quad (2)$$
$$contract(id7, [parties: [Y, p_c]]), \quad (3)$$

$$good\_faith(GF, [agt: p_c, obj: Falsity]). \quad (4)$$

At this point, no goal in the goal-list can be proved, since no fact in Fig. 4 is concerned with any contract or its falsity. However, goal (2) is partly resolved by the legal theory rule in Fig. 7. Thus, our deductive engine replaces (2) with the sub-goal list:

$$contract(Ctrct1) \quad (5)$$

$$repr\_of\_ctrct(R1, [obj: Ctrct1, atr: ReprCts1]) \quad (6)$$

$$soa\_of\_ctrct(S1, [obj: Ctrct1, atr: \text{RealCts1}]) \quad (7)$$

$$RepreCts1 \neq RealCts1 \quad (8)$$

Consequently, the goal list becomes [(1), (5), (6), (7), (8), (3), (4)].

Now no rule in the rule base is applicable to any goal in the goal list. Thus we try to generalize a goal: *contract*(*Ctrct*1, [*parties*: [p_a, *Y*]]). Generalizing the predicate *contract* by

$$contract \rightarrow juristic\_act$$

atom (1) is replaced with

$$juristic\_act(Ctrct1, [parties: [p_a, Y]]) \quad (9)$$

At this point, we generally have three choices:

Choice1: to continue to generalize the goal (9).

Choice2: to try generalization for other predicate instances: goals (2) or (3).

Choice3: to verify the generalized instance *juristic_act*1.

The first choice might lead to so called over-generalization, and should, therefore, be delayed. The second choice is also possible, at least theoretically. However, it is desirable to apply Choice3 before Choice2, because generalization with less variable-binding generally increases the size of the search space. For these reasons, we try to prove goal (9). However, this also fails because the current case is not a juristic act. We therefore turn to Choice2. Similarly, goal (3) is then replaced with

$$juristic\_act(Ctrct2, [parties: [Y, p_c]]), \quad (10)$$

By repeation, completely similar generalizations *contract*, *ctrct_falsity*, *repr_of_ctrct* and *soa_of_ctrct* are generalized to *lawful_act*, *falsity*, *repr* and *soa*, respectively, to give the goal-list:

lawful_act(*Ctrct*1, [*parties*: [p_a, *Y*]]),
lawful_act(*Ctrct*1, −),
repr(_, [*obj*: *Ctrct*1, *atr*: *ReprCts*]),
soa(_, [*obj*: *Ctrct*1, *atr*: *RealCts*]),
ReprCts ≠ *RealCts*,
lawful_act(*id*7, [*parties*: [*Y*, p_c]]),
good_faith(*GF*, [*agt*: p_c, obj: *Falsity*])

Putting *Ctrct*1 = *id*3, *Y* = p_b, *ReprCts* = p_b, *RealCts* = p_a, the first six goals can be deduced from the rule base as well as the set of facts in the present case. It is also clear that *falsity*(_, [*obj*: *id*3]]) cannot be deduced from the facts indexed by p_c. Hence the last goal is also derived by the meta-inference rule on negative predicates.

3.6 Some Comments on Conflict in Generalizing Rules

We have described various types of knowledge used for interpreting legal rules analogically and a fundamental mechanism to perform interpretation. Both generalization and analogy give us possible ways of interpreting legal rules. As long as our generalization and analogy are based on a legal conceptual hierarchy with which most lawyers agree, the hypothesis produced by our system will be approved. However, our knowledge base might contain several rules to generalize legal rules in different ways. Some lawyers may agree with some generalizations, while others may disagree. Thus there may exist conflicts in the generalization of rules. From a computational point of view, these problems of conflicts can be considered as problems in the nondeterministic choice of generalizations. Roughly speaking, we have two ways in mind for coping with this problem.

The first is to introduce semantic constraints by which some inappropriate generalizations are rejected because of inconsistency with the constraints. This is also a well-known approach for checking the appropriateness of generalizations. It can be implemented, in principle, in an ATMS-like truth maintenance system. However, it is true that we cannot decide what constraints are needed before reasoning. Therefore, the truth maintenance system is used only for checking the appropriateness of hypotheses.

The second way of coping with the problem of the nondeterministic choice of generalizations is now being investigated by one of the authors. The key idea is to find past cases in which the same or similar generalizations were used. This is thus a case-based justification of generalization and analogy. According to this approach, no new generalizations are found by our system. However, the system can generate a case-based explanation showing why it chooses a particular generalization to interpret legal rules. A forthcoming paper will clarify the idea further.

4. Case-Based System for Statutes

In this section, we introduce HELIC-II as an example of a case-based system (Fig. 15) [9, 10]. HELIC-II consists of two inference engines—the rule-based engine and the case-based engine—and draws conclusions on the parallel computer PIM (Parallel Inference Machine). A legal reasoning system for the penal code has been developed by using HELIC-II. Given a new case, HELIC-II generates all possible legal consequences and their explanations by referring to old cases and the penal code.

4.1 The Penal Code

In the penal code, general provisions and definitions for all crimes are given as legal rules. Though they seem to be defined strictly, the interpretation of some legal

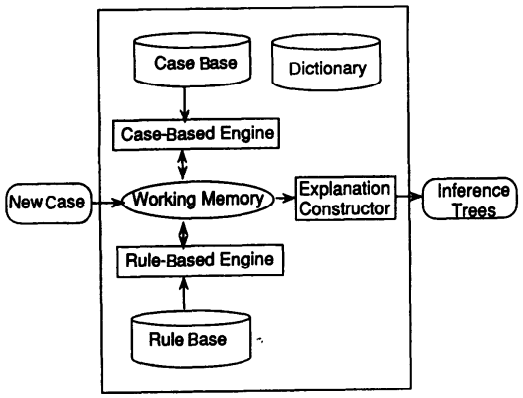

Fig. 15 HELIC-II.

Article 38 (Intent, negligence):
Clause 1 No person shall be punished for an act performed without criminal intent.
Clause 2 When a person who committed a crime was not, at the time of its commission, aware that he was committing a crime graver than the one he thought it to be, he shall not be dealt with in accordance with the crime he committed.

Fig. 16 Article 38 of the Japanese Penal Code.

concepts and the matching of conditions cause difficult problems. The following are examples of the problems involved.

- (Intention) The prosecutor has to prove the existence of criminal intention because of Article 38 (Fig. 16).
  For example, let us take the following problems:
  A person X wanted to kill Y and fired a gun at him, but the bullet hit another person Z and killed him.
  In this case, X did not intend to kill Z. Deciding X's crime is problematic because X will insist that he should only be punished for the crime of manslaughter caused by negligence.
- (Causality) The prosecutor has to prove the causality between action and result. For example, let us take the following problem:
  A person X fired a gun at Y, and Y was shot in his hand. Since Y had a heart condition, Y died of a heart attack caused by the shock of the injury.
  In this case, X's action was not the main reason for Y's death. However, X may be punished for the crime of homicide. Legal causality is different from physical causality. As legal causality is similar to the concept of liability and its interpretation is affected by various knowledge, it is impossible to give necessary and sufficient conditions of "causality."

4.2 Representation of Cases

The rule-based engine of HELIC-II refers to the legal rules of the penal code and outputs possible crimes by deductive reasoning. The legal rules are represented by about 200 clauses. The LHS (left hand side) of a clause is a condition, and the RHS (right hand side) is a legal concept. Though LHSs contain abstract predicates such as negligence, intention, and causality, the facts of a new case do not contain these predicates. Therefore, at first, the case-based engine begins to reason and generates such predicates by referring to similar old cases. These predicates are sent to working memory, and the rule-based engine then draws conclusions by using these data.

An old case is represented as a *situation* and several *case rules*. The *situation* consists of the events of the case. A *case rule* is a fragment of the arguments of both sides [bra]. The argument of one side is represented as a chain of case rules.

(1) Situation

A situation is represented by a set of objects/events and their temporal relations. The following is an example of the description of a situation. Each object/event has the form

⟨*name of concept*⟩ (⟨*object/event identifier*⟩, ⟨*list of slots*⟩)

where *slot* is a pair of an *attribute* and its *value*. The value of a slot is an atom, an identifier of another object/event, or a variable.

problem(*trafficAccident*112, "*example*",
before(*dinner*1, *drive*1), *during*(*accident*1, *drive*1). .]).
dinner(*dinner*1, [*agent*=*john*, *place*=*maxims*]).
drive(*drive*1, [*agent*=*john*, *car*=*toyota*1]).
accident(*accident*1, [*agent*=*john*]).
...
caused(*cause*1, [*cause*=*accident*1, *effect*=*injury*1]).
person(*john*, [*sex*=*male*, *age*=30]).
person(*mary*, [*sex*=*female*]).
injury(*injury*1, [*agent*=*mary*]).
restaurant(*maxims*, [*rank*=5*stars*]).
car(*type*=*sportsCar*]).
...

The meaning of this example is that the case "*trafficaccident*112" consists of three events such as "*dinner*1," "*drive*1," and "*accident*1." "*drive*1" occurred after "*dinner*1," and "*accident*1" occurred during "*drive*1." The object "*dinner*1" is an instance of "dinner", and is acted on by "*john*" in "*maxim's*," and so on.

A situation represented by this language is translated into a semantic network as in Fig. 17.

(2) Case Rules

A case rule has the form

⟨*rule name*⟩ (⟨*comment*⟩, ⟨*list of rule information*⟩,
[*A*1, *A*2, . . . , *Ai*]—>[*B*1, *B*2, . . . , *Bk*]).

The LHS of a case rule is a subset of a situation and

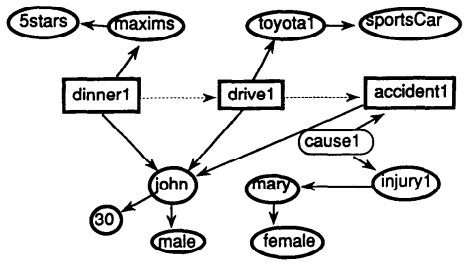

Fig. 17 Semantic Network of a Situation.

the RHS is a consequence insisted on by one side. The following is a simplified example of the case rule of the prosecutor.

*rule*001("*example*",
[*article*=218, *insisted*=*prosecutor*, *result*=*won*],
[*drive*(*drive*1, [*agent*=*john*/*important*,
car=*toyota*1/*trivial*]),
accident(*accident*1, [*agent*=*john*/*important*]),
caused(*id*1, [*cause*=*accident*1/*important*,
effect=*injury*1/*important*]),
injury(*injury*1, [*agent*=*mary*/*important*])]
person(*john*, [*sex*=*male*/*trivial*]),
person(*mary*, [*sex*=*female*/*trivial*])]
→
[*responsibility*(*Resp*, [*agent*=*john*,
object=*care*1]),
takeCare(*care*1, [*agent*=john, *object*=*mary*])]).

The meaning of ths case rule is "in the case that John caused a traffic accident while driving Toyotal and Mary was injured by the accident, John has a responsibility to take care of her."

We store not only the case rules employed by the judge but also case rules rejected in the court, because defeated case rules can still, sometimes, provide hints for formulating arguments.

(3) Conceptual Hierarchy

Conceptual hierarchy is defined in the dictionary as

⟨*concept*⟩(⟨*super concept*⟩, ⟨*list of attribute*⟩)

The following is an example:

object(*creature*, [ ]).
creature(*person*, [*age*, *sex*]).
person(*person*, [ ]).
person(*infant*, [ ]).
creature(*lion*, [ ]).
action(*drive*, [*agent*, *car*, *destination*]).

The similarity between different concepts is defined by their distance in the conceptional hierarchy.

4.3 Reasoning by Similarity

The role of the case-based engine is to compare a new case with case rules from old cases and to draw legal consequences by generating new case rules. The reasoning consists of the following two stages (Fig. 18).

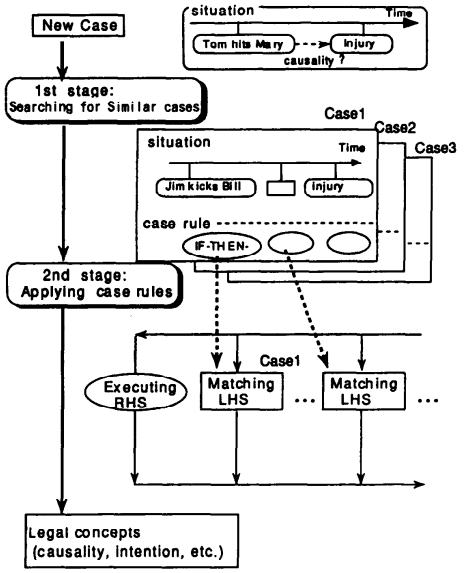

Fig. 18 Reasoning by a Case-Based Engine.

- (First Stage) The engine searches for similar situations in a case base. The similarity between two situations is measured by comparing the sequences of events in each situation. If two sequences have a long subsequence of common events, the situations are recognized as similar. To obtain a common subsequence, each event is generalized by using a conceptual hierarchy. Therefore, even if two events are different, they may be recognized as similar if their upper concept is the same.
- (Second Stage) For each case selected in the first stage, the engine searches for case rules whose LHS partially matches the new case, and applies the rules. The degree of matching between the LHS and a new case is evaluated by mapping the LHS to a new case and counting the number of mapped links of the semantic network. If the number is higher than a threshold, the LHS is judged as similar to the new case.

A set of generated legal concepts may contain conflict (such as intention against non-intention) because they are generated by the opinions of both prosecutors and defendants. Therefore, the output of the case-based engine is classified into several combinations of legal concepts that do not contain conflicts, and statutes are applied separately in the rule-based engine.

4.4 Example of Penal Code

(1) Mary's Case

We will show how the following case is solved by HELIC-II. We selected this case from the bar examination of lawyers in Japan.

Mary's Case:
On a cold winter's day, Mary abandoned her son Tom on the street because she was very poor. Tom was just 4 months old. Jim found Tom crying on the street, and started to drive Tom by car to a police station. However, he caused an accident on the way to the police station. Tom was injured. Jim thought that Tom had died of the accident, and left Tom on the street. Tom froze to death.

The problem is to decide the crimes of Mary and Jim, describing the reasons and theories employed.

The difficult issues in this case are as follows. The critical point is that the injury is not the main cause of Tom's death.

- Causality between Mary's action and Tom's death.
- Causality between Jim's accident and Tom's death.

The situation of Mary's case is represented as follows:

problem("*mary's case*", . . .)
abandon(*aba*1, [*agent*=*mary*, *object*=*tom*]).
pickup(*pic*2, [*agent*=*jim*, *object*=*tom*]).
. . .
trafficAccident(*acc*1, [*agent*=*jim*]).
. . .
person(*mary*, [*sex*=*female*]).
baby(*tom*, [*sex*=*male*]).
. . .
injury(*injury*1, [*agent*=*tom*]).
caused(*cause*2, [*cause*=*acc*1, *effect*=*injury*1]).
. . .
injury(*death*9, [*agent*=*tom*]).

As there are no definite rules for the causality, it is impossible to judge this case by using a rule-based system. Therefore, HELIC-II searches for old cases in which similar problems were handled.

For example, the following is part of an actual case handled by the Superme Court of Japan:

Jane's Case:
Jane wanted to kill Dick, so she strangled him while he was sleeping. Dick only lost consciousness, but Jane thought that he was dead. She took him to the seashore, and left him there. He inhaled sand and suffocated to death.

In the court, the prosecutor insisted that Jane should be punished for the crime of homicide, for the following reasons:

- Strangling and taking to the seashore should be considered the stream of actions involved in the homicide. Therefore, it is evident that there was an intention to kill Dick and causality between her actions and Dick's death.
- Otherwise, there is causality between strangling and Dick's death though some unpredicted event intervened.

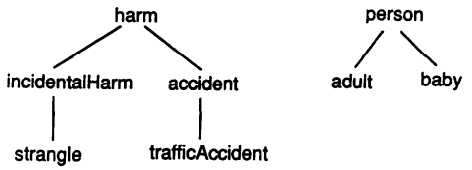

Fig. 19 Conceptual hierarchy.

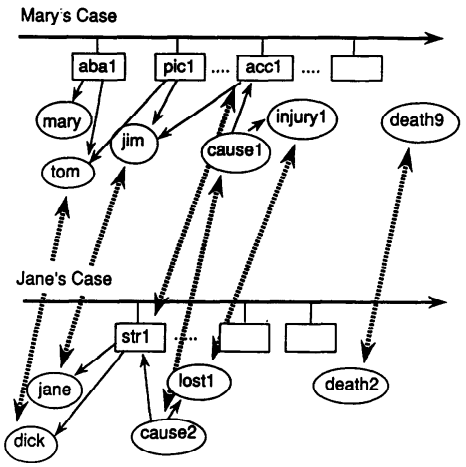

Fig. 20 Mapping a case rule to Mary's case.

On the other hand, Jane insisted her actions did not satisfy the condition of the crime of homicide, for the following reasons:

- Strangling and taking to the seashore should be considered different actions. There is no causality between strangling and Dick's death, and there was no intention to kill him when taking him to the seashore. Therefore, the actions do not satisfy the conditions for homicide.

Let Jane's opinion be represented by the following case rule:

*rule*002("*example*", [*article*=218, *insist*=*defendant*, *result*=*won*],
[*strangle*(*str*1, [*agent*=*jane*/*trivial*, *object*=*dick*/*trivial*]),
intention(*int*1, [*agent*=*jane*/*trivial*, *object*=*act*1/*important*, *goal*=*death*1/*important*]),
death(*death*1, [*agent*=*dick*/*trivial*]),
caused(*cause*1, [*cause*=*act*1/*important*, *effect*=*lost*1/*important*]),
lostConsciousness(*lost*1, [*agent*=*dick*/*important*]),
misunderstand(*mis*1, [*agent*=*jane*/*trivial*, *fact*=*lost*1/*important*, *recognition*=*death*2])],
. . .
→
[~*causality*(*cause*1, [*cause*=*act*1, *effect*=*death*3])]).

After the case-based engine selects Jane's case in the

first stage, it evaluates the similarity between the LHS of the above rule and Mary's case. For example, "strangling" and "trafficAccident" are lower concepts of "harm," and "baby" is a lower concept of "person" in the conceptional hierarchy (Fig. 19). Therefore, *suf1*, *jane*, and *dick* of the LHS are respectively recognized as similar to *acc1*, *jim*, and *tom* of Mary's case. In this way, as most of the LHS of this rule can be mapped to Mary's case (Fig. 20), the case-based engine generated "$\sim causality(cause=acc1, effect=death9)$." Using this concept, the rule-based engine draws the conclusion that "Jim should be punished for the crime of death caused by negligence in the conduct of business."

Through this method, HELIC-II generates crimes such as the crime of abandonment by a person responsible, resulting in death, the crime of death caused by negligence, and so on. Some parts of the reasoning employ the prosecutor's opinion, while other parts employ the defendant's opinion.

Usually, matching between semantic networks takes a long time. However, by using the 64 processors of the parallel computer PIM, the case-based engine draws conclusions more than 50 times faster than a single processor.

4.5 Other Case-Based Systems

HELIC-II represents explanations of old cases by semantic networks and case rules with the reasoning executed by partial matching of semantic networks. This mechanism is similar to the reasoning mechanism of GREBE [14], which also draws conclusions by rule-based reasoning and case-based reasoning. The inference mechanism of HELIC-II was made simpler than that of GREBE in order to obtain high-performance parallel inference.

In this subsection, we introduce HYPO as an example of another case-based system for common law [11]. HYPO compares and contrasts several cases and constructs arguments by dynamically changing the focal characters of cases.

The legal knowledge of HYPO is given as a *case knowledge base* (CKB) and a library of *dimensions*.

A dimension identifies features that affect final conclusions. It defines the magnitudes of the selected features, and it has information on the effect on both parties. A dimension is a similar concept to a slot value. In trade secret law, by analyzing judicial precedents, Rissland selected about 30 dimensions as key features. For example, "competitive-advantage" and "bribe-employee" are pro-plaintiff dimensions and "disclose-secrets" and "generally-known" are pro-defendant ones. When a new case is given, HYPO checks which dimensions hold with what magnitudes, and they are compared to old cases.

A case is represented as a frame that contains related dimensions, a final conclusion (pro-plaintiff or pro-defendant), and other information. As a case does not

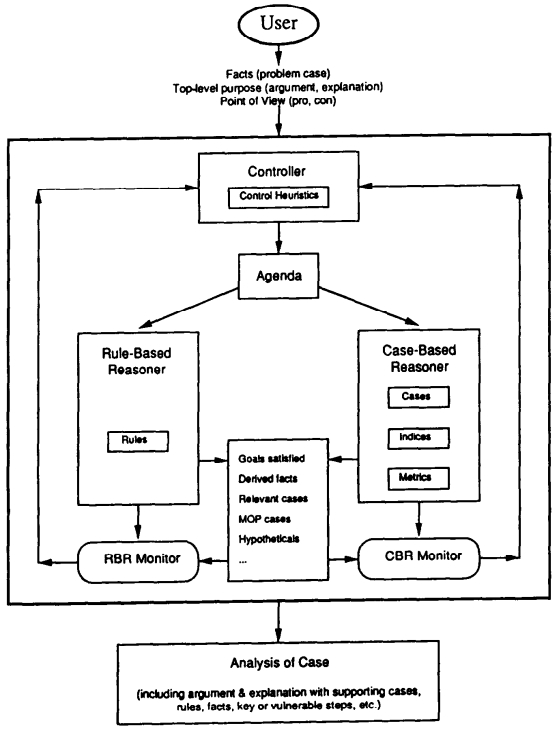

Fig. 21 CABARET.

contain explanations that combine initial facts and final conclusions, reasoning is performed by using a set of dimensions. The similarities between two cases are judged by common dimensions and the differences are judged by uncommon dimensions. An old case is more similar to a new case than other cases if the old case shares more dimensions.

Given a new case, HYPO can generate a 3-ply argument as follows:

1. (Citation) HYPO selects the most relevant cases that support the new case and are most similar to it.
2. (Responding) HYPO distinguishes such cases in terms of uncommon dimensions and selects more relevant counter-examples in the claim lattice.
3. (Rebuttal) HYPO rebuts the response by distinguishing the cases again.

CABARET [12] is a hybrid legal reasoning system that consists of the case-based reasoning component and the rule-based component. The reasoning mechanism of HYPO is used in its case-based reasoning component. The two reasoning modules of CABARET are controlled by a controller, which contains heuristics to provide information for an agenda (Fig. 21).

A case of HELIC-II contains detail reasoning steps from facts to final conclusions. By using a conceptual hierarchy and case rules, a situation can be modified at

various abstraction levels. Therefore, it can generate several explanations (interpretations). The problems involved in HELIC-II are as follows:

- It is troublesome to represent cases. As the situation often consists of many actions, many temporal relations between actions and the mental status of agents, the description of a situation often becomes very complex.
- It is difficult to preserve the stability of the representation. The representation of the situation varies with the developer of the case base.
- It take a significant amount of time to compare semantic networks.

On the other hand, in HYPO, it is very easy to describe old cases and compare cases. However, HYPO uses only one-level case abstraction, and its reasoning ability is limited. Furthermore, the performance of HYPO depends on the quality of the dimensions. If there are insufficient dimensions, it will fail to generate useful arguments. Therefore, to decide on the set of dimensions, old cases should be analyzed in detail.

HELIC-II and CABARET are hybrid systems that consist of two inference engines. CABARET has a controller that controls a rule-based module and a case-based module. A controller has heuristics to manage reasoning steps, and complex control is realized. In the case of HELIC-II, there is no controller, because HELIC-II was developed on a parallel computer (PIM) and two inference engines reason in parallel. The final inference trees are constructed in an explanation constructor.

5. Conclusion

We explained the knowledge used by lawyers, and introduced an example of the representation of legal knowledge. One feature of legal knowledge is that it is too abstract to be applied directly to specific cases. This is not a defect, because it provides legal knowledge with flexibility and stability.

Since such features of legal reasoning are not realized by a simple rule-based system, we described two systems that interpret legal rules. LES is a rule-based system that can treat interpretation by analogical reasoning. In the field of civil code, the reasoning mechanism of LES will be useful to compensate for the defects of the rule system.

HELIC-II uses case-based reasoning to complement the rule-based system. Since old cases contain much information, a case-based approach is more powerful for handling open-textured concepts. However, a case-based approach cannot generate interpretations if there are no similar old cases, and there is no definite way to gather typical good cases and to extract useful information. Since many cases must be stored in the case base, some powerful tools are needed to construct a case base so that we can develop practical systems.

The reasoning processes of LES and HELIC-II are useful mechanisms for realizing one aspect of legal reasoning. However, they are not sufficient to explain actual reasoning steps. To develop a powerful legal reasoning system, it is necessary to develop more higher-order inference mechanisms.

References

1. TANAKA, H. *Introduction to the study of positive law (in Japanese)*, University of Tokyo Press, 1974.
2. HARAGUCHI, M. *A form of analogy as an abductive inference, In Proc. 2nd Workshop on Algorithmic Learning Theory*, Japanese Society for Artificial Intelligence (1991), 266–274.
3. MUGGLETON, S. and BUNTINE, W. *Machine invention of first-order predicates by inverting resolution, In Proc. Workshop on Machine Learning* (1988), 339–352.
4. MUGGLETON, S. *Inductive logic programming, in Proc. 1st Workshop on Algorithmic Learning Theory* (1990), 42–66.
5. YOSHINO, H., HARAGUCHI, M., KAGAYAMA, S. and MATSUMURA, Y. *Foundation of the systematization of analogy in law* (in Japanese), *In Proc. National Conference of Japanese Society for Artificial Intelligence* (1991), 219–222.
6. YOSHINO, H. *Legal expert system LES-2, in Springer Lecture Notes in Computer Science, Logic Programming '86* (1987), 34–45.
7. AOUMI, Z. *Introduction to the Philosophy of Law* (in Japanese), Koubunndou, 1989.
8. GASYUU, H. *Analogy in law* (in Japanese), unpublished lecture notes, Legal Expert Systems Association, Meiji Gakuinn Univ., Tokyo, 1986.
9. NITTA, K. et al. Experimental Legal Reasoning System on a Parallel Inference Machine, PPAI Workshop of 12th IJCAI (1991).
10. NITTA, K. et al. HELIC-II: A Legal Reasoning System on a Parallel Inference Machine, To appear in Proc. Int. Conf. on Fifth Generation Computer Systems (1992).
11. RISSLAND, E. L. et al. A Case-Based System for Trade Secrets Law, International Conference on Artificial Intelligence and Law (1987).
12. RISSLAND, E. L. et al. Interpreting Statutory Predicates, International Conference on Artificial Intelligence and Law (1989).
13. SARTOR, G. The structure of Norm Conditions and Nonmonotonic Reasoning in Law, International Conference on Artificial Intelligence and Law (1991).
14. BRANTING, L. K. Representing and Reusing Explanations of Legal Precedents, International Conference on Artificial Intelligence and Law (1989).
15. SANDERS, K. Planning in an Open-Textured Domain: A Thesis Proposal, Technical Report CS-91-08, Brown University (1991).
16. SANDERS, K. Representing and reasoning about open-textured predicates, International Conference on Artificial Intelligence and Law (1991).

(Received December 3, 1991)