new HELIC-II: A Software Tool for Legal Reasoning

Katsumi Nitta, Masato Shibasaki, Tsuyoshi Sakata,
Takahiro Yamaji, Wang Xianchang, Hiroshi Ohsaki (JIPDEC),
Satoshi Tojo (MRI) and Iwao Kokubo (MRI)

Institute for New Generation Computer Technology
4-28, Mita 1-chome, Minato-ku, Tokyo 108, Japan
nitta@icot.or.jp

Abstract

The *new HELIC-II* is a software tool for legal reasoning. It consists of two functions - argumentation function and debating function. Argumentation function is realized by a typed logic programming language with generalization of rules and defeasible reasoning based on priority of rules. Debating function is realized by meta knowledge which controls the argumentation function.

This paper introduces overview of the *new HELIC-II* system. We show how legal knowledge is represented in the *new HELIC-II* illustrated by presenting the example of an actual criminal case.

1 Introduction

In the field of AI and Law, many models of legal reasoning have been proposed. However, most of them are focused on one aspect of legal reasoning such as making arguments or selecting an argument or debating.

For example, we have already developed a legal reasoning system HELIC-II [Nitta(a)] in the FGCS project. This system (old HELIC-II) is a hybrid system which consists of two inference engines - a rule base reasoner and a case base reasoner. We showed the effectiveness of the hybrid architecture by presenting solutions to several criminal cases.

However, the old HELIC-II has the following problems.

(1) Although it can generate many alternative arguments, there is no function to select the best one. (2) It was implemented in a parallel logic programming language KL1 and runs only on the parallel inference machine PIM. Therefore, it lacks portability.

To resolve these problems, we started development of the *new HELIC-II* at the FGCS Follow-on project.

The research target of the *new HELIC-II* is to propose a unified model of legal reasoning, and to develop a portable software tool based on the model.

This system has two functions - argumentation function and debating function. And the argumentation function consists of two functions such as making and selecting argument.

In this paper, we give a brief introduction to the key concepts of the *new HELIC-II* system. In Section Two, we give overview of the *new HELIC-II*, and in Sections Three and Four, we introduce the argumentation function and the debating function.

2 Overview of the system

In this section, we introduce two functions of the *new HELIC-II*.

2.1 Argumentation Function

When the judge solves legal problems in the suit, he observes the facts, makes a conclusion and generates an argument to support the conclusion. If he can make more than one conclusions, he must compare each argument and select the best one. We call this thinking process as argumentation function and modeled it by two modules - making arguments and selecting arguments (Fig.1).

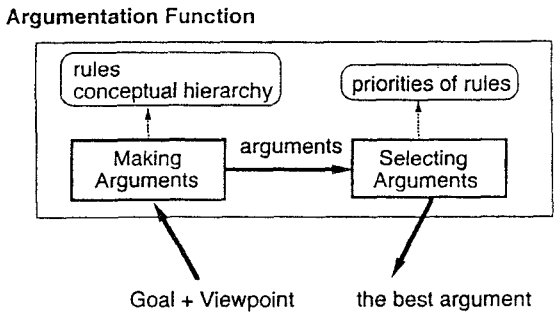

Figure 1: Argumentation Module

When the user input a desired conclusion (goal) and his viewpoint, "the making arguments module" generates all arguments which achieve the goal, all counter arguments for each argument, all counter counter arguments for each argument, and so on. Then, "the selecting argument module" selects the best one based on user's viewpoint. User's viewpoint is compiled into priorities of rules, and the strength of arguments is measured by them.

Much research has been conducted into making arguments by combining a rule base reasoner and a case base reasoner. For example, GREBE [Branting], CABARET [Rissland] and HELIC-II [Nitta(a)] have hybrid architectures and make arguments for a given case. Figure 2 is the architecture of the HELIC-II.

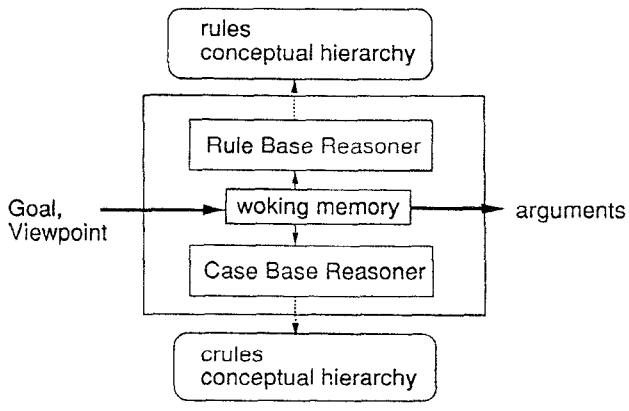

Figure 2: RBR and CBR

On the otherhand, the research of [Prakken] and [Sartor] looked at the selection of arguments based on nonmonotonic reasoning.

We unified parts of results of these researches and we designed a new language to realize the argumentation function. This language is a typed logic programming language with generalization of rules and defeasible reasoning. We will explain the language in Section Three.

2.2 Debating Function

Though an argumentation function simulates judge's thinking process, it is not sufficient to simulate attorney's thinking process. For example, the argumentation function assumes that there is only one viewpoint (priority of rules) and one knowledge base. However, in the actual case, the prosecution and the defense may have different viewpoints and different knowledge. Therefore, there is a debate in which both parties insists that their argument is more suitable than opponent's.

To simulate attorney's thinking process, we need another function - debating function - which generates subgoals, enhances viewpoints during debate process and controls the argumantation function (Fig. 3).

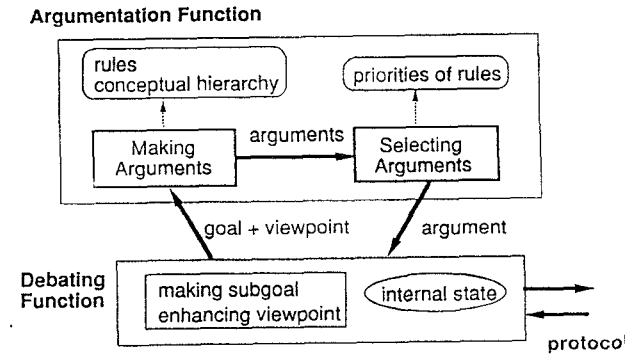

Figure 3: Debating Function

We modeled the debate strategy as follows.

1. Initially, two parties have different rules and different viewpoints. They don't know what rules and viewpoints the opposite side may have.

2. When a user inputs an initial goal, the debate starts.

   Both parties present their claims to each other (Fig. 4). There are several claims as follows. (1) To make a new argument for a given goal. (2)

To find issue goals in the arguments posed by the opponent. (3) To decide if an argument is defeated by a counter argument or not. (4) To enhance the viewpoint of this side to make the argument of this side to defeat the opponent's argument.

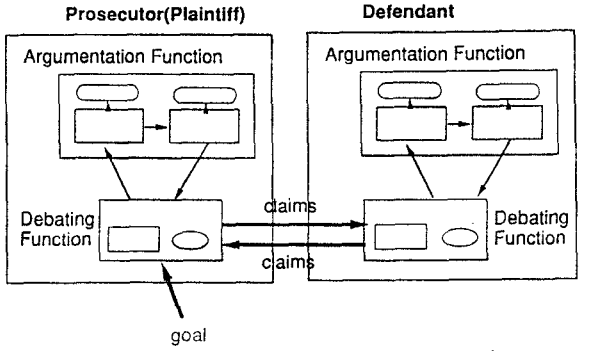

Figure 4: Debate between prosecution and defense

3. The defeat relation between an argument and its counter argument is defined by priorities of rules which appear in an argument or a counter argument. If both parties have the same priority $rule_1 > rule_2$, we can decide the defeat relation between an argument and its counter argument. However, if the prosecution insists $rule_1 > rule_2$, and the defendant insists $rule_2 > rule_1$, we cannot decide the defeat relation.

4. During the debate process, the current viewpoint of each side may be enhanced by attaching new priority relations of criteria of value.

Already research into the debate model has been conducted by [Rissland] [Ashley] [Loui] [Gordon]. Our model is different from theirs in that we focused on the difference of viewpoints of both sides and combined this into the argumentation function.

3 Argumentation Function and KR Language

We developed a knowledge representation language which realizes the argumentation function. In this section, we explain several important concepts of our language.

3.1 Knowledge Representation

3.1.1 Type

The primary components of knowledge representation are an "object", an "event" and "property." We call classes of objects, events and property as "object type", "event type" and "property type."

Between object types, we have partial order relations ($>_o$), between event types, we have partial order relations ($>_e$), and between property type, we have partial order relations ($>_p$). They construct lattices, respectively.

Let T_o, T_e and T_p be a set of object types, event types and property types. There are special type $\top$ and $\bot$, and for any $O \in T_o$, $E \in T_e$ and $P \in T_p$, the following relations hold.

$$\top >_o O,\ O >_o \bot,\ \top >_e E,$$
$$E >_e \bot,\ \top >_p P,\ P >_p \bot$$

For any event type and property type, we can define another type with negation ($\neg$).

3.1.2 Terms

We introduce two terms - ψterm and Hterm. A ψterm is used to define an object, and an Hterm is used to define an event or a property.

(a) ψterm

A ψterm is defined as an object type, or a structure constructed by a root symbol and a list of object labels as follows [Ait-Kaci].

$$P/person[age \Rightarrow 20, last\_name \Rightarrow X/string,$$
$$parent \Rightarrow person[last\_name \Rightarrow X]]$$

Here, a *root symbol* (*person*) is an object type symbol, and an *object label* consists of an object label symbol (*age*, *last_name*, *parent*) and its value (20, $X/string$, $person[last\_name \Rightarrow X]$). We can attach an object tag symbol (P, X) before an object type. To the same object tag symbols, the same ψterms are substituted.

Semantics of ψterm : Let U be a universe of ψterms and let I_o be an interpretation of ψterm, then for any $O \in T_o$, $I_o[O]$ is a subset of U, and the following relations hold.

$$I_o[\top] = U,\ I_o[\bot] = \{\ \}$$
$$\forall O_1,\ O_2 \in T_o\ O_1 <_o O_2 \rightarrow I_o[O_1] \subset I_o[O_2]$$

A *label* is a function from U to U. Interpretation of ψterms ($\Psi_1 = O[l \Rightarrow t]$ and $\Psi_2 = O[l_1 \Rightarrow t_1, ..., l_n \Rightarrow t_n]$) is defined as follows.

$$I_o[\Psi_1] = \{x \in I_o[O] \mid \exists y \in I_o[t], I_o[l](x) = y\}$$
$$I_o[\Psi_2] = \bigcap I_o[O[l_i \Rightarrow t_i]]$$

If two ψterms ($\Psi_1 = O_1[l_1 = A_1, ...]$, $\Psi_2 = O_2[l'_1 = A'_1, ...]$) satisfy following three conditions, then $\Psi_1 \preceq_o \Psi_2$ holds.

(1) $S_1 \leq_o S_2$,

(2) if $l = A'$ appears in the label of Ψ_2, then $l = A$ appears in the label of Ψ_1, and $A \leq_o A'$ holds,

(3) constraints by tags of Ψ_2 are satisfied by Ψ_1.

If $\Psi_1 \preceq_o \Psi_2$ then $I_o[\Psi_1] \subseteq I_o[\Psi_2]$ holds.

(b) Hterm

An Hterm is defined as a structure consisting of a root symbol and a list of event label or property label as follows.

$A/watch(agent = X/person,$
$object = hit(agent = Y/person,$
$object = Z/person[sex \Rightarrow male],$
$cause = \#fight))$

Here, a root symbol is an event type or a property type. A label value may be a ψterm or an Hterm. "*#fight*" is an Hterm tag, and it is replaced by another Hterm. Hterm tags appearing in facts are preceded by #, and those appearing in rules are preceded by @.

Let E_1 and E_2 be elements of T_e. If two Hterms ($H_1 = E_1[l_1 = A_1, ...]$, $H_2 = E_2[l'_1 = A'_1, ...]$) satisfy the following four conditions, then $H_1 \preceq_e H_2$ holds.

(1) $E_1 \leq_e E_2$,

(2) if $l = A'$ appears in the label of H_2, $l = A$ appears in the label of H_1 and A and A' are Hterms, then $A \preceq_e A'$ holds or $A \preceq_p A'$ holds,

(3) if $l = A'$ appears in the label of H_2, $l = A$ appears in the label of H_1 and A and A' are ψterms, then $A \preceq_o A'$ holds,

(4) constraints by tags of H_2 are satisfied by H_1.

Let P_1 and P_2 be elements of T_p. If two Hterms ($H_1 = P_1[l_1 = A_1, ...]$, $H_2 = P_2[l'_1 = A'_1, ...]$) satisfy the following four conditions, then $H_1 \preceq_p H_2$ holds.

(1) $P_1 \leq_p P_2$,

(2) if $l = A$ appears in the label of H_1, $l = A'$ appears in the label of H_2 and A and A' are Hterms, then $A \preceq_e A'$ or $A \preceq_p A'$ holds,

(3) if $l = A$ appears in the label of H_1, $l = A'$ appears in the label of H_2 and A and A' are ψterms, then $A \preceq_o A'$ holds,

(4) constraints by tags of H_2 are satisfied by H_1.

Semantics of Hterm : Interpretation of Hterm consists of interpretation of ψterm I_o and a mapping π from Hterm to $\{true, false\}$. π satisfies the following two conditions.

(1) $\pi(\top) = true$, $\pi(\bot) = false$,

(2) for two Hterms H_1 and H_2, if $H_1 \preceq_e H_2$ or $H_1 \preceq_p H_2$, then
$\pi(H1) = true \rightarrow \pi(H2) = true$
holds.

3.1.3 Rules

A rule consists of a unit name, consequence part and condition part as follows.

$U :: A \leftarrow B_1, B_2, ..., B_n$

Here, A is an Hterm, and B_i is an Hterm or an Hterms preceded by "*not*." "*not*" means "negation as failure".

Some rules may be generalized when applied to a new case. We distinguish such rules from others, and call them "*Crules*."

3.1.4 Priority of Rules

We can define a unit name for not only a rule but a group of rules. For example, let r_1, r_2, r_3 be unit names of three rules. Then,

$r_0 := \{r_1, r_2, r_3\}$

defines a new unit name "r_0" which represents these three rules.

"Priority of rules" is defined as a priority name and a set of priority relations of unit names.

$p_1 := \{r_1 > r_2\}$

Moreover, we can define "priority of priority." It is defined as an identification name and a set of priority relations of priority names.

$view_1 := \{p_1 > p_2, p_3 > p_4\}$

3.2 Inference

3.2.1 Unification

(a) Unification of ψterm

Let X_i and Y_i be ψterms and let θ be defined as follows.

$\theta = \{X_1/Y_1, X_2/Y_2, ..., X_n/Y_n\}$

If θ satisfies the following two conditions, then θ is a *substitution* of ψterm.

(1) Tags appearing in Y_i are different from tags appearing in X_j,

(2) $X_i \succ_o Y_i$.

The result of substituting ψ for θ is represented as $\psi\theta$, and $\psi\theta \succ_o \theta$ holds.

For two ψterms ψ_1, ψ_2, if there exists a substitution θ which satisfies $\psi_1\theta = \psi_2\theta \neq \bot$, then ψ_1 and ψ_2 are called *unifiable*.

(b) Unification of Hterm

Let X_i and Y_i be event types or property types and let η be defined as follows.

$$\eta = \{X_1/Y_1, X_2/Y_2, ..., X_n/Y_n\}$$

If η satisfies the following two conditions, then η is a *substitution* of Hterm.

(1) Tags appearing in Y_i are different from tags appearing in X_j,

(2) $X_i \succ_e Y_i$ or $X_i \succ_p Y_i$.

For two Hterms H_1, H_2, if there exists a substitution η which satisfies $H_1\eta = H_2\eta \neq \bot$, then H_1 and H_2 are called *unifiable*.

3.2.2 SLD Resolution and an argument

(a) SLD Resolution

Let G_i be the following goals

$$\leftarrow A_1, .., A_k, ..., A_n.$$

and C_i be

$$A \leftarrow B_1, ..., B_m. \in Rules$$

and θ be an *mgu* (most general unifier) which satisfies $A\theta \preceq_e A_k\theta$, then the following G_{i+1} is derived from G_i and C_i.

$$\leftarrow (A_1, .., A_{k-1}, B_1, .., B_m, A_{k+1}, ..., A_n)\theta.$$

(b) Argument

An argument of a goal "G" is a set of instantiated rules which draws "G."

For example, let concepts, rules and facts be as follows.

$tom < person, bill < person, person < object$

$shot\_a\_gun < act, act < event$

$r1 :: punishable(agent = X) \leftarrow$
$act(agent = X/person,$
$object = Y/person) \mid @act,$
$crime(agent = X, goal = homicide).$

$r2 :: crime(agent = X, goal = homicide) \leftarrow$
$act(agent = X/person,$
$object = Y/person) \mid @act,$
$is\_homicide(a\_object = @act),$
$neq(X, Y).$

$r3 :: is\_homicide(a\_object = @act) \leftarrow$
$act(agent = X/person,$
$object = Y/person) \mid @act,$
$with\_intent(a\_object = @act, goal = homicide),$
$died(agent = Y) \mid @death,$
$causality(a\_object = @act, goal = @death).$

$r4 :: -punishable(agent = X) \leftarrow$
$act(agent = X/person[age \Rightarrow [0..13]],$
$object = Y/person) \mid @act.$

$f1 :: shot\_a\_gun(agent = tom,$
$object = bill) \mid \#act.$
$f2 :: with\_intent(a\_object = \#act,$
$goal = homicide).$
$f3 :: died(agent = bill) \mid \#death.$
$f4 :: causality(a\_object = \#act, goal = \#death).$

Following is an argument for a goal "$crime(agent = tom, goal = homicide)$."

$Arg(crime(agent = tom, goal = homicide)) =$
$\{crime(agent = tom, goal = homicide) \leftarrow$
$shot\_a\_gun(agent = tom, object = bill),$
$is\_homicide(a\_object =$
$shot\_a\_gun(agent = tom, object = bill)),$
$neq(tom, bill).$
$is\_homicide(a\_object =$
$shot\_a\_gun(agent = tom, object = bill)) \leftarrow$
$shot\_a\_gun(agent = tom, object = bill),$
$with\_intent(a\_object =$
$shot\_a\_gun(agent = tom, object = bill),$
$goal = homicide),$
$died(agent = bill),$
$causality(a\_object =$
$shot\_a\_gun(agent = tom, object = bill),$
$goal = died(agent = bill)).$
$shot\_a\_gun(agent = tom, object = bill).$
$with\_intent(a\_object =$
$shot\_a\_gun(agent = tom, object = bill),$
$goal = homicide).$
$died(agent = bill).$
$causality(a\_object =$
$shot\_a\_gun(agent = tom, object = bill),$
$goal = died(agent = bill)). \}$

3.2.3 Defeasible reasoning based on priority of rules

(a) Defeat Relation

Let A and B be two Hterms.

If A and $\neg B$ are unifiable, or if $\neg A$ and B are unifiable, then A and B are said to be a *contradiction*. For two arguments $Arg(A)$ and $Arg(B)$, if A and B cause a contradiction, we say "$Arg(A)$ attacks $Arg(B)$" and "$Arg(B)$ attacks $Arg(A)$". For any argument $Arg(G)$, if $Arg(A) \subset Arg(G)$ and $Arg(A)$ attacks $Arg(B)$, then we say $Arg(B)$ is a *counter argument* of $Arg(G)$. In this case, we call $-B$ as *issue point* in $Arg(G)$.

Let $Arg(B)$ be a counter argument of $Arg(G)$, and let r_2 and r_1 be top default rules including $Arg(B)$ and $Arg(-B)$. If one of the following conditions holds, then we say $Arg(B)$ *defeats* $Arg(G)$ (Fig.5).

(1) r_2 is only one default rule included in $Arg(B)$,

(2) for any sub argument of $Arg(B)$ which doesn't include r_2, it is a *justified argument.*

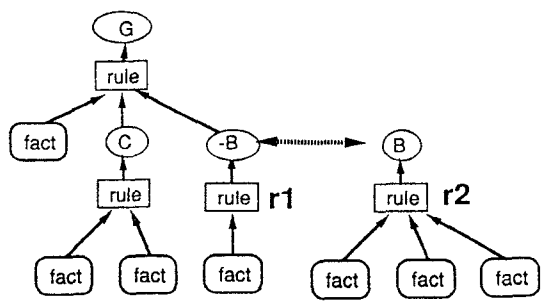

Figure 5: Defeat relation

An argument can be classified into three categories - a *defeated* argument, a *justified* argument and a *merely plausible* argument [Sartor].
A defeated argument is an argument which is defeated by some counter argument. A justified argument is an argument which defeats any counter arguments. A merely plausible argument is an argument which is neither a defeated one nor a justified one. Justified arguments and merely plausible ones are called *plausible* ones.

(b) Example of defeasible reasoning

Let consider rules and facts in the previous page again.
We can make an argument which supports "$punishable(agent = tom)$" by using rules r_1, r_2 and r_3. However, for example, if Tom is 12 years old, then we can make two arguments which conflict each other as follows.

$Arg(punishable(agent = tom[age \Rightarrow 12]))$
by r_1, r_2 and r_3
$Arg(-punishable(agent = tom[age \Rightarrow 12]))$
by r_4

If r_1 has priority over r_4, then the former argument is stronger. On the contrary, if r_4 has priority over r_1, then the former argument is defeated by the latter. Therefore, to select the best argument, we must input not only a goal but priority of rules. The priority of rules is closely related to the personal viewpoint.

(c) Search Level

To decide an argument is justified or not, we must consider its counter arguments, its counter-counter arguments, and so on. If we calculate these arguments completely, the search space is very large. However, in the debate process between two parties, complete calculation is useless, because priority of rules may change during debate. Therefore, we prepared a mechanism to limit the search space. For example, if we set a search level to one, then the system calculate the defeat relation of counter arguments, but it doesn't consider counter-counter arguments.

3.2.4 Generalization of Crules

(a) Generalization

While ordinal rules are used as they are, "Crules" may be generalized before they are used, and their condition parts are matched based on similarity. For example, let consider following concepts

$hit <_e do\_violence$, $kick <_e do\_violence$,
$fainted <_e suppressed$, $feared <_e suppressed$,
$mary <_o girl$, $jane <_o girl$,
$tom <_o boy$, $bill <_o boy$,
$girl <_o person$, $boy <_o person$
$watch <_o property$, $purse <_o property$

and following Crule.

$r5 :: -criminal\_intent(agent = X,$
$object = \#hit, goal = robbery) \leftarrow$
$hit(agent = X/tom, object = Y/mary) \mid \#hit,$
$criminal\_intent(agent = X,$
$object = \#hit, goal = violence),$
$fainted(agent = Y),$
$take\_away(agent = X,$
$object = watch[owner \Rightarrow Y]).$

This Crule is extracted from judicial precedents. Its meaning is "Tom hit Mary to damage her, and she was fainted. He took away her watch while she was fainted. In this case, the judge thought that Tom didn't have the criminal intent of robbery."

This Crule may be generalized as follows.

$r6 :: -criminal\_intent(agent = X,$
$\quad object = \#hit, goal = robbery) \leftarrow$
$\quad do\_violence(agent = X/boy,$
$\quad\quad object = Y/girl) \mid \#hit,$
$\quad criminal\_intent(agent = X,$
$\quad\quad object = \#hit, goal = violence),$
$\quad suppressed(agent = Y),$
$\quad take\_away(agent = X,$
$\quad\quad object = property[owner \Rightarrow Y]).$

The meaning of a generalized rule is "a boy did violence to a girl to damage her, and she was suppressed. He took away her property while she was suppressed. In this case, the judge thought that the boy didn't have the criminal intent of robbery."

Here, if we have following facts,

$kick(agent = bill, object = jane) \mid \#kick.$
$criminal\_intent(agent = bill, object = \#kick,$
$\quad goal = violence).$
$feared(agent = jane).$
$take\_away(agent = bill,$
$\quad object = purse[owner \Rightarrow jane]).$

then we can make an argument which supports a goal

$-criminal\_intent(agent = bill, object = \#hit,$
$\quad goal = robbery)$

by applying this generalized rule.

Crules are used to represent judicial precedents, theories of interpretation and some statutory rules. The generalization of Crules corresponds to application of precedents to similar new cases and corresponds to widening interpretation of statutory rules.

However, if a Crule is generalized without any constraint, the condition part will be meaningless because it may be generalized to $\top$. Therefore, we prepared some control mechanism. By describing the upper limit of generalization in Crules, we can restrict the level of generalization.

(b) Similarity based matching

If we represent judicial precedents by Crules, their condition parts become a set of facts. In this case, all conditions are not always important. Even if some conditions are not satisfied, if most important conditions are satisfied, then such Crules should be applied.

To realize similarity based matching of Crules, we define a similarity level as follows.

$$similarity = \frac{\sum_i weight_i * satisfied\_condition_i}{\sum_i weight_i * condition_i}$$

If a new case satisfies the condition part of Crule completely, the similarity level becomes 1.0, and if a new case doesn't satisfy the condition part at all, the similarity level becomes 0. If this value is more than predefined threshold, we regard the condition part is satisfied.

Generally, not important condition of a Crule can be generalized more than important condition. To discriminate not important information from others, we introduce the notation "!." This information is used as weighting value to calculate the similarity value.

3.3 Description of Legal Knowledge

In this section, we show how legal knowledge is represented in our language.

3.3.1 Concepts

We must define concepts (objects, events, properties) as a type hierarchy (a conceptual dictionary). Legal knowledge such as facts, rules, theories, judicial precedents and criteria for value judgment should be represented using concepts in the dictionary.

3.3.2 Statutory rules and Legal Theories

As most legal rules take the form of "if - then - unless rules", they are easily represented as rules. The following are examples of articles 36 and 199 of Japanese Penal Code.

$p36 :: punishable(a\_object = @act)$
$\quad \leftarrow act(agent = X/person,$
$\quad\quad object = Y/person) \mid @act,$
$\quad crime(a\_object = @act, goal = Z/crime),$
$\quad not\ self\_defense(a\_object = @act).$
$p199 :: crime(a\_object = @act, goal = homicide)$
$\quad \leftarrow act(agent = X/person,$
$\quad\quad object = Y/person) \mid @act,$
$\quad is\_homicide(a\_object = @act),$
$\quad neq(X, Y).$

As predicates which appear in legal rules are vague, supplementary rules are needed to make the meaning of vague concepts clear. Legal theories concerning interpretation of legal rules are also supplementary rules. For example, concerning criminal intent of theft, there are three theories (interpretations).

Theory A: If a person takes away something, he has criminal intent of theft.

Theory B: Even if a person takes away something, if he doesn't intend to pretend as he is the owner of it, he doesn't have criminal intent of theft.

Theory C: Even if a person takes away something, if he doesn't intend to use it himself, he doesn't have criminal intent of theft.

These theories are also represented as rules of our language. As different lawyers may adopt different interpretations, there may exist rules which conflict each other.

3.3.3 Judicial precedents

Precedents contain information such as facts, a final decision, arguments of both sides, and the argument of the judge. Arguments included in precedents consist of several levels of rules. While rules appearing near the conclusion consist of more general conditions, rules near facts consist of concrete conditions.

Though rules of the former level are applied as they are, rules of the latter level are applied after they are generalized. Therefore, the latter level rules are represented as Crules.

Following is an example of judicial precedents concerning criminal intent of theft. " Tom hid Bill's car in order to interfere Bill's date. Though Tom didn't intend to use it himself, the judge decided that Tom had criminal intent of theft." It is represented as a Crule.

3.3.4 Criteria for value judgment and viewpoint

If there are several interpretations of a statutory rule, and if their conclusions conflict each other, a lawyer must select the best one. When a lawyer select the best interpretation, he compares each interpretation based on his viewpoint. We describe a personal viewpoint as a priority relation between criteria of value.

The followings are examples of criteria of value.

property: Private property must be protected.

balance: Conditions of crimes should be harmonized and well balanced.

flexibility: Legal rules should be interpreted with flexibility.

principle: Widening interpretation should not be allowed in the criminal law.

Among them, "flexibility" and "principle" often conflict. Therefore, we can define meta level criteria as follows.

focus_on_flexibility := $\{flexibility > principle\}$.

Three theories concerning criminal intent of theft which we explained before are classified by these criteria.

$property := \{'TheoryA'\}$.
$balance := \{'TheoryB', 'TheoryC'\}$.
$flexibility := \{'TheoryC'\}$.

A *viewpoint* is a priority relation of criteria of value as follows.

$view1 := \{focus\_on\_flexibility > balance\}$

A viewpoint is compiled as priority of rules, and it is used for defeasible reasoning.

Different people have different viewpoints, and different people have different best arguments.

4 Debating Function and Claims

4.1 Six Claims

Debate is conducted between the prosecution and the defense by exchanging claims each other. The user can take part of prosecution side or defense side, and the system takes the other side.

At first, the user inputs a knowledge base which consists of a dictionary and rules, and facts of a new case. Then, he sets initial viewpoints for both sides. When he inputs the initial goal to the prosecution side, the goal is sent to the argumentation module and it makes an argument (an original argument) for the goal, and shows it with a list of issue points. Then, the debate starts.

There are six claims which both sides can issue. Both sides select one of them by turns.

- **issue:** Among a list of issue points, select one as current goal. Then, this goal is sent to the argumentation module and an argument for the goal is generated. This argument is a counter argument of original argument.

- **pose:** When an argument is generated and the viewpoint is enhanced, send the argument to the opponent.

- **justify:** Enhance the viewpoint of this side in order to make the counter argument stronger than original argument.

- **notify:** Notify the opponent that the argument of this side is stronger than opponent's one.

- **cancel:** As the argument of this side is weaker than opponent's one, give up the current argument.
- **finish:** Notify the opponent that there is no effective counter argument.

The debate process is shown in the figure 6. As both sides finds an issue point from opposite side's current argument, the debate progresses in depth first, and if there is no effective counter argument, then backtrack occurs and another issue point which is already listed up is selected as a current issue point.

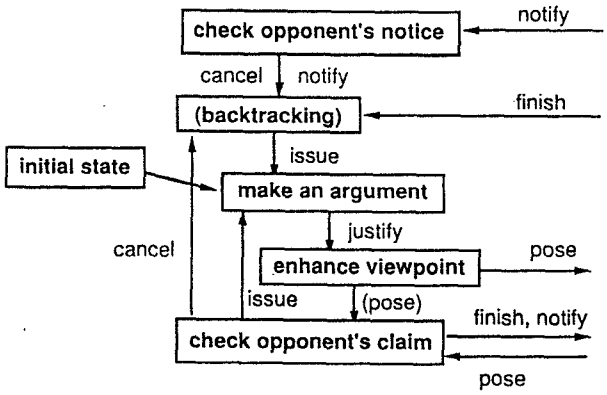

Figure 6: Debate strategy

4.2 Example of Debate

We will show an example of the debate process. We selected this example from the lawyers' examination.

(1) A new case (Mary's case)
"Mary had hated Jane for a long time and wanted to hurt her. Mary waited Jane in the street, and hit her in the face. Jane had a bad fall, and she lost consciousness. Mary thought Jane was dead. Then Mary took away Jane's handbag in order to make other people that Jane was killed during robbery. Mary threw the handbag in the river next day. Which crime should Mary be punished for?"

(2) Issues in Mary's case
This case contains several hard issues concerning interpretation of the Japanese Penal Code. One issue is whether hitting Jane is the crime of inflicting an injury or the crime of violence.

The second issue is whether taking the handbag is punishable as the crime of robbery, the crime of theft, or the crime of embezzlement. How to evaluate Mary's intent affects the conclusion.

The third issue is whether abandoning a handbag is punishable as the crime of damage to property.

Concerning the above issues, there are several theories and precedents, and different lawyers support different interpretations depending on their viewpoints.

(3) Example of debate process
As *new HELIC-II* doesn't have a mechanism which represents arguments in natural languages, it just shows arguments in the form of inference trees. The following is an explanation of one of the debate processes which the *new HELIC-II* generates. In this example, (P) is the prosecution claim and (D) is the defense claim.

1. (P) According to "Theory A", taking away a handbag is the crime of theft.
2. (D) According to "Theory B", to prove the existence of criminal intent of theft, intent to deceive another into believing that he or she is the owner of the property must be shown. As Mary didn't have this intent, her action is not the crime of theft.
3. (P) From the viewpoint of protecting property, "Theory A" is relevant.
4. (D) From the viewpoint of balancing between conditions of theft and embezzlement, "Theory B" is more relevant.
5. (P) It was a long time from taking away the handbag to throwing it in the river. By common sense rule, Mary intended to use it herself. Therefore, Mary had criminal intent of theft by "Theory C."
6. (D) "Theory B" is relevant from the viewpoint of principle of criminal law.
7. (P) As regards the flexibility of law, "Theory C" is more relevant.
8. (D) Mary threw away the handbag after all. By a common sense rule, she had no intention to use it. Therefore, she didn't have intent of theft.

5 Conclusion

We have introduced the *new HELIC-II* system. Here is a summary of our evaluation of the system.

(1) As a software tool for legal reasoning
The reasoning model of the *new HELIC-II* contains important components of legal reasoning such as generating arguments, value judgement and debate strategy. It can also treat various legal knowledge such as statutes, precedents, legal theories, criteria of value evaluation, personal viewpoints and debate strategies. We can thus use the *new HELIC-II* to investigate many aspects of legal reasoning.

As the *new HELIC-II* is implemented on a parallel logic programming language KLIC, it is compiled into C programs and it runs on the Unix and DOS/V environments.

(2) As a language for legal knowledge
We have developed a typed logic programming language for legal knowledge. Theoretically, it contains many inretesting topics.

Among functions of our language, rule generalization mechanism is a very powerful generator of the interpretation of legal rules. However, as the current generalization mechanism is too simple. We need more function to control generalization.

(3) As a debate model
One of the important features of the *new HELIC-II* is that the personal viewpoint plays an important role during debate. Though this model is useful to explain the process of value judgment, current description of the viewpoint is too simple. We need another mechanism to represent the viewpoint in addition to priority between criteria of value judgment.

(4) Future works
Though FGCS Follow-on Project finished in March, the first author will continue this research in Electrotechnical Laboratory (email: nitta@etl.go.jp). Following is a list of future works.

- Flexible control mechanism for generalization.
- Theoretical research concerning generalization and analogical reasoning.
- User's language to describe debate strategy.
- Extension of defeasible reasoning which can calculate defeat relation effectively.
- Developing Legal knowledge base with lawyers.

Acknowledgements

In conducting this research, discussions with professor Hajime Yoshino (Meijigakuin University) and members of his legal reasoning project was very useful. In addition, we would like to express our appreciation for their suggestions to Dr. Thomas Gordon (GMD), Dr. Ronald Loui (Washington University), and Dr. Giovanni Sartor (CIRFID).

References

[Ait-Kaci] H. Ait-Kaci and R. Nasr, LOGIN: A Logic Programming Language with Built-In Inheritance, In *Journal of Logic Programming*, pp. 185-215, 1986.

[Ashley] K. Ashley, Modeling Legal Argument: Reasoning with Cases and Hypotheticals, MIT Press, 1990.

[Branting] K. Branting : Integrating Rules and Precedents for Classification and Explanation: Automatic Legal Analysis, ph D. thesis, Univ. Texas, 1991.

[EDR] User's manual, EDR, 1984.

[Gordon] T. Gordon : The Pleading Game - An Artificial Intelligence Model of Procedural Justice, ph D. Thesis, GMD,1993.

[Loui] R. Loui : Computing Specificity, Research Report, WUCS-92-46, Washington Univ., 1992.

[Nitta(a)] K. Nitta, et al. : HELIC-II: A Legal Reasoning System on Parallel Inference Machine, Proc. Int. Conf. on FGCS92, 1992. pp. 1115 - 1124.

[Nitta(b)] K. Nitta, et al. : A Computational Model for Trial Reasoning, Proc. Int. Conf. on Artificial Intelligence and Law, 1993. pp. 20 - 29.

[Prakken] H. Prakken : Logical Tools for Modeling Legal Argument, Ph D. thesis, Vrije Universiteit, 1993.

[Rissland] E.L. Rissland et al. : A Case-Based System for Trade Secrets Law, Proc. Int. Conf. on Artificial Intelligence and Law, 1987. pp. 60 - 66.

[Sartor] G. Sartor : A Simple Computational Model for Nonmonotonic and Adversarial Legal Reasoning, Proc. Int. Conf. on AI and Law, 1993. pp. 192 - 201.