TM-1301

Knowledge Representation of NEW HELIC-II

by

K. Nitta, M. Shibasaki, T. Sakata, T. Yamaji,
H. Ohsaki (JIPDEC), S. Tojo (MRI),
I. Kokubo (MRI) & T. Suzuki (MRI)

June, 1994

Knowledge Representation of New HELIC-II

Katsumi Nitta, Masato Shibasaki, Tsuyoshi Sakata, Takahiro Yamaji
Hiroshi Ohsaki, Satoshi Tojo, Iwao Kokubo, Takayuki Suzuki

Institute for New Generation Computer Technology (ICOT)

nitta@icot.or.jp

1 Introduction

Since 1991, we have been developing a legal reasoning system, HELIC-II [Nitta 92], on the parallel inference machine PIM [Taki 92], developed by ICOT. This system consists of two inference engines - a rule-based engine and a case-based engine - as well as three knowledge sources - a rule base of legal rules, a case base of old cases and a conceptual dictionary. HELIC-II can solve legal reasoning problems by referring to statutory rules and old cases, and generate possible legal consequences and their explanations.

Although HELIC-II is a powerful reasoning system, its inference mechanism is not sufficient as a general legal reasoning model. Based on our experience with HELIC-II, we commenced our work in a new version of HELIC-II last year. Our goal for the new HELIC-II is to develop a model of general legal reasoning, and to realize it as a software tool in logic programming language KLIC [Chikayama 89].

Especially, we focused our attention on the reasoning process of argument and on the role of standpoint and viewpoint during argument, because we feel that the cognitive aspect is vital to create a viable legal reasoning model.

As an example of legal reasoning, we selected the reasoning process followed by an attorney at law. An attorney's task is characterized by the following features.

- The goal is provided by the client. The main task is to create an argument to support the goal. To establish an argument, he draws on his knowledge of statutes, theories, old cases and facts, as well as applying commonsense. He may modify the existent rules and apply them to facts.

- He argues against an opposing party in court. Issues arise mainly from the existence of vague predicates and conflicts in rules. There are several ways to attack the opposite side such as denying an opposite side's argument, making a new counter argument and finding another evidence.

As the reasoning process that the attorney goes through an argument covers several aspects of legal reasoning, a reasoning model of an attorney becomes a general framework of legal reasoning. However, to develop a legal reasoning system which realize this framework, we need a knowledge representation language which can treat statutes, old cases and meta rules. Therefore, developing a new language is one of important subjects of our research.

In this paper, we introduce the functions of the new HELIC-II and its knowledge representation language NHL [1].

[1]Temporarily, we are referring to our language as NHL, standing "New HELIC-II Language."

2 Overview of the new HELIC-II

2.1 Functions of the new HELIC-II

There are three target functions of the new version of HELIC-II as follows.

The first creates an argument for pursuing a given goal. This function consists of devising arguments to achieve a goal, listing up counter arguments, and to finding strong, favorable arguments. Therefore, the function is a kind of theorem prover, corresponding to the first step undertaken by the prosecutor in court.

The second function creates a counter argument to defeat the other party's argument. This is realized by changing the relative priorities of rules or by changing facts. This corresponds to the work of the prosecutor attorney as they debate.

The third function predicts the judgement in court. This corresponds to the work of the judge who has his own viewpoints.

2.2 Three layer reasoning model

The model of new HELIC-II consists of three layers (Fig. 1).

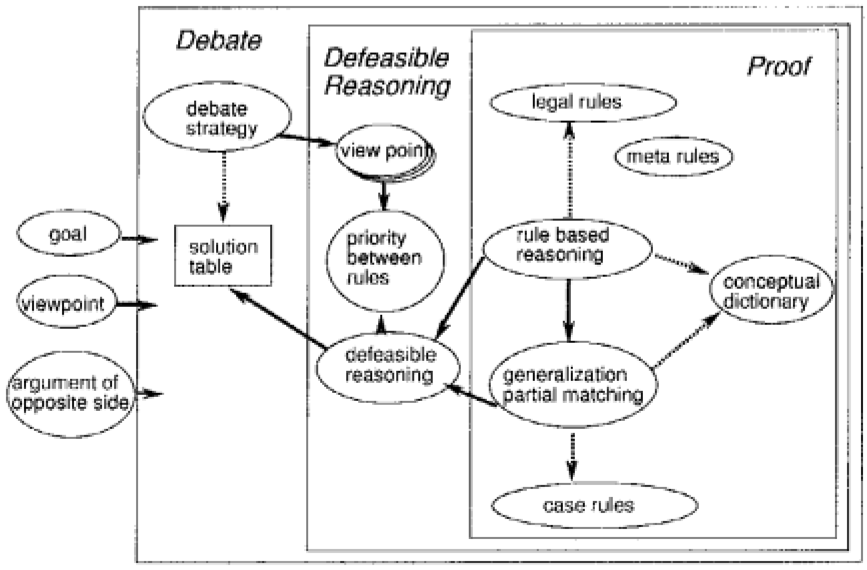


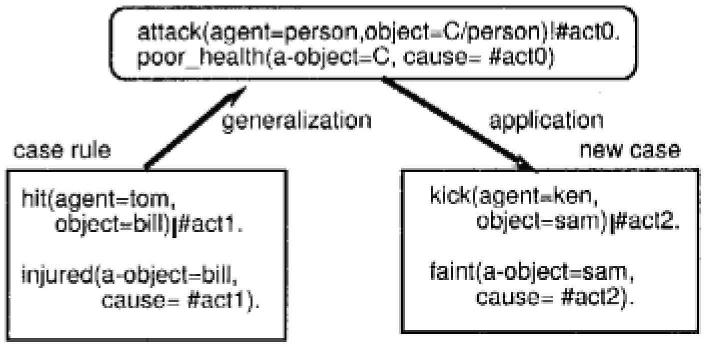
Figure 1: Three layers of legal reasoning

1. **Proof layer (Theorem Proving):**
This layer refers to statute, old cases, legal theories, commonsense knowledge and a conceptual dictionary, and generates an argument for pursuing a given goal and given facts (evidence). To model this layer, we assume following assumptions.

- All concepts which appear in facts and rules are given as a conceptual dictionary.
- Statutes, legal theories and induced rules extracted from old cases are represented as a set of *legal rules.*
- Judgements in old cases are represented as *case rules.* A case rule differs from legal rules in that its condition part is matched to facts based on similarity [Branting 89] [Nitta 92].
- Except for the conceptual dictionary and some commonsense knowledge, legal rules and case rules may contradict.

Given a goal, this layer tries to prove the goal by applying legal rules, case rules and meta rules in the following order.

- Apply legal rules.
  This step is realized by deductive reasoning of legal rules.
- Apply case rules.
  This step is realized by reasoning by analogy. Before applying a case rule, its condition part is replaced by more general condition and if main part of condition is satisfied by new case, then the rule is applied. For example, let a case rule be "In the case that Tom hit Bill and Bill was injured, Tom was punished by crime of violence." The condition part of this case rule consists of two facts: "Tom hit Bill" and "Bill was injured by the action." These facts are generalized as "A person attacked a person" and "the victim was in poor health by the action." If a new case contains following facts: "Ken kicked Sam" and "Sam was fainted by the action", a new case was considered to be similar to old case because a new case satisfies the generalized conditions (Fig. 2).



```
attack(agent=person,object=C/person)|#act0.
poor_health(a-object=C, cause= #act0)
```

```
hit(agent=tom,
    object=bill)|#act1.

injured(a-object=bill,
        cause= #act1).
```

```
kick(agent=ken,
     object=sam)|#act2.

faint(a-object=sam,
      cause= #act2).
```

```
crime_of_violence(agent=B)
          <- A/hit(agent=B/tom, object=C/bill)|#act1,
             D/injured(a-object=C, cause=#act1)
             | { limit(A,attack),.....}.
```

Figure 2: Similarity matching

- Modify legal rules by generalization using meta rules, and then apply the modified rules.
  The third step corresponds to the *interpretation of legal rules.*
- Modify initial facts by meta rules.
  This step corresponds to find new evidence.

2. **Defeasible reasoning layer (Reasoning based on viewpoint):**
   If the first layer generates conclusions which conflict each other, then lawyers have to select more reasonable one. This is realized by nonmonotonic reasoning techniques. This layer contains a module that handles defeasible reasoning and viewpoints. Our defeasible reasoning is based on the concept of priority between rules [Sartor 93], [Prakken 93]. If two defeasible rules result in a conflict, a rule that has priority over another rule is employed. A viewpoint consists of the relative priorities of standpoints, while a standpoint consists of the relative priorities of legal rules, case rules and meta rules. For example, "*Lex Superior:* Case rules of the supreme court have priority over those of the district court", "*Lex Posterior:* Newer case rules have priority over older ones" are standpoints. And "Lex Posterior has priority over Lex Superior" is viewpoint. One viewpoint corresponds to a set of priorities between legal rules and case rules. If two parties take different viewpoints, they will generate different plausible arguments.

3. **Debate layer:**
   Each attorney has his own viewpoint. However, his viewpoint isn't always complete but is partial priorities between standpoints. During debate in the court, an attorney will make up with his viewpoint. This layer contains a module that finds a counter argument to defeat the other party's argument by reinforcing the current viewpoint.

   To model this argument process, we assumed the following.

   - Both parties have the same knowledge bases, these being a set of legal rules, a set of case rules and a conceptual dictionary. Both parties have different initial viewpoints.
   - Both parties are unaware of the other party's viewpoint.

   Debate is realized by modifying the argument each party makes against the other. During argument, the viewpoints possessed by both sides increases.

3 A knowledge representation language *NHL*

3.1 Features of the language

We designed a knowledge representation language *NHL* to develop a legal reasoning system based on our previous analysis.

NHL covers the first and second layers of the legal reasoning model (Fig.1). The features of *NHL* are as follows.

(1) Types and terms:
NHL uses "types" to represent basic concepts. Two kinds of terms are constructed, using types to distinguish absolute knowledge from defeasible knowledge. While a *ψ-term* is considered to be absolute knowledge, *H-term* is considered to be defeasible.

(2) Defeasible reasoning.
Legal rules and case rules may draw conflicting conclusions. To resolve this contradiction, *NHL* uses defeasible reasoning based on the priorities of rules. The priorities of rules are considered to be a viewpoint or standpoint.

(3) Two kind of negations.
Legal rules contains default rules and exception rules. *NHL* allows the use of both "negation as conflict" and "negation as failure."

(4) Reasoning by analogy.
To make use of old cases, *NHL* has mechanism to control reasoning by analogy.

3.2 Overview of *NHL*

NHL is an extension of the logic progaramming language "LOGIN" [Ait-Kaci 86][Smolka 89], by implicit case, negation as failure, defeasible reasoning and reasoning by analogy.

(1) Type

Type is a set of objects, and it consists of ***verb-type*** and ***noun-type***. The partial order relation of two types, $A <_T B$, means that A is a subsumed by B. $\top$ is the set of all objects, while $\perp$ is a null set (Fig. 3).

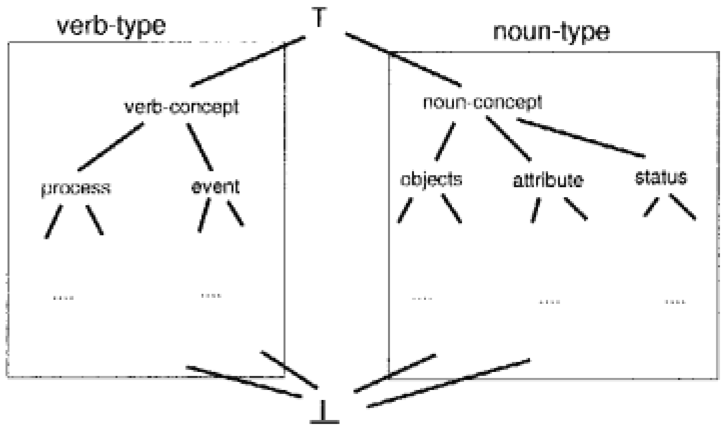


Figure 3: Verb-type and noun-type

Type definition consists of an "attribute definition" and a "subsumption relation" between types.

attribute definition:
$Type - [l_1 : f_1, ..., l_n : f_n]$
subsumption relation:
$Type_i <_T Type_j$
typed variable:
$X/Type$
$X/Type[l_1 : f_1, ..., l_n : f_n]$
($Type$, $Type_i$ and $Type_j$ are type symbols, l_i is a label, X_i is a variable,
f_i is a typed variable, and $<_T$ is a subsumption relation.
In typed variable, "$X/$" can be omitted.)

For example, let "person", "integer", "country", "japan" and "japanese" be type symbols. And let "age", "parent" and "nationality" be labels. Then, the following is an example of type definitions.

person−[age:integer, parent:person, nationality:country].

japanese=[nationality:japan].
japanese $<_T$ person.
japan $<_T$ country.

(2) Terms

Using types, we define different kinds of terms - *modified ψ-term* and *H-term*.

Modified ψ-term is a modification of $\psi - term$ introduced by [Ait-Kaci 86]. In this paper, we call $modified\psi - term$ as $\psi - term$. It takes following form.

X
$Qtype$
$Qtype[l_1 \Rightarrow \psi_1, l_2 \Rightarrow \psi_2, ..., \psi_n \Rightarrow t_n]$
$X/Qtype[l_1 \Rightarrow \psi_1, l_2 \Rightarrow \psi_2, ..., \psi_n \Rightarrow t_n]$
(X is a variable, *Qtype* is a quantified type, l_i is a label, ψ_i is a ψ-term.)

Quantified type takes the following form and it is used to represent an anonymous instance.

some Type
("*some*" is an annotation and "*Type*" is a type.)

"*some Type*" is a type which is subsumed by "*Type*" and there is no type which is subsumed by "*some Type*." For example,

$some\ tree[owner = tom]$

means some tree owned by Tom. It is subsumed by

$tree[owner = tom]$

and there is no type between "$some\ tree[owner = tom]$" and "$\perp$" (Fig. 4).

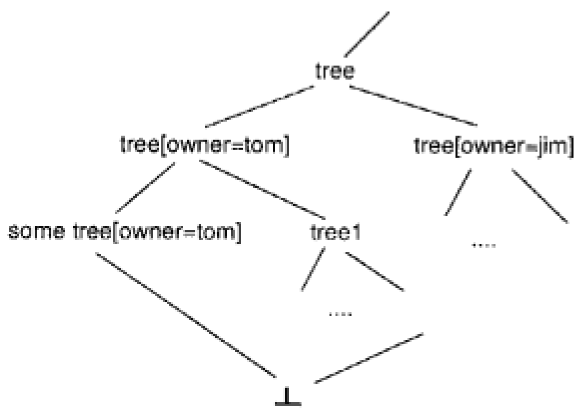


Figure 4: Qualified type

H-term takes the following forms.

$$v(c_1 = q_1, c_2 = q_2, ..., c_n = q_n)$$
$$\neg\, v(c_1 = q_1, c_2 = q_2, ..., c_n = q_n)$$

(v is a ψ-term whose top is a verb-type, c_i is a *case symbol* and q_i is an *H-term* or *ψ-term*.)

Here, we will explain the relation between *H-term* and *ψ-term*. Let the *h-term_type* be defined as follows.

$$h\text{-}term\_type{=}[verb : verb\_type, c_1 : X_1, ..., c_n : X_n]$$

Then, the H-term

$$v(c_1 = q_1, c_2 = q_2, ..., c_n = q_n)$$

is considered as a syntax sugar of the following ψ-term, if none of cases are omitted.

$$h\text{-}term\_type[verb \Rightarrow some\ v, c_1 \Rightarrow q_1, ..., c_n \Rightarrow q_n]$$

As H-term is a syntax sugar of ψ-term, a set of case symbols is a subset of a set of labels.

If some cases are omitted in an H-term of a rule head, we consider such cases have skolem constants. On the contrary, if some cases are omitted in an H-term in condition part, then such case values are considered to be variables. As facts are rules whose condition part is omitted, following two facts are treated as different events.

$$hit(agent = tom, object = jim, place = house1)$$
$$hit(agent = tom, object = jim)$$

In the case of *ψ-term*, omitted labels are considered as variables. Therefore, for example, following relation holds.

$$hit[agent \Rightarrow tom, object \Rightarrow jim, place \Rightarrow house1]$$
$$<_T \ hit[agent \Rightarrow tom, object \Rightarrow jim]$$

Another difference between ψ-term and H-term is that ψ-term cannot denied while H-term has negative form. Negative form of H-term is treated using *positive form* [Gelfond 90].

Let *positive-h-term* be defined as follows.

$$positive\text{-}h\text{-}term\_type{=}[verb : v, c_1 : X_1, ..., c_n : X_n]$$

Then, the positive forms of following two H-terms

$$h\text{-}term\_type[verb : v, c_1 : X_1, ..., c_n : X_n]$$
$$\neg\ h\text{-}term\_type[verb : v, c_1 : X_1, ..., c_n : X_n]$$

are defined as follows, respectively.

$$h\text{-}term\_type[verb : v, c_1 : X_1, ..., c_n : X_n]$$
$$positive\text{-}h\text{-}term\_type[verb : v, c_1 : X_1, ..., c_n : X_n]$$

To prove a given goal is satisfied, H-terms are replaced by their positive forms and proof procedure is conducted on the positive forms. If both

$$h\text{-}term\_type[verb : v, c_1 : X_1, ..., c_n : X_n]$$

and

$$positive\text{-}h\text{-}term\_type[verb : v, c_1 : X'_1, ..., c_n : X'_n]$$

appear in the same proof and they are unifiable, then the proof contains contradiction in it.

(3) H-term identifier

Other than a variable which appear in ψ-term, *NHL* uses an H-term-variable.

A *H-term-variable* is a symbol whose first character is "@". It appears just after H-term and used as H-term identifier. In H-term denoted by H-term-variable, omitted cases are treated as they have universal quantified variables even if the H-term appears in the rule head.

For example, if "place" and "time" are implicit cases of "hit", then the following rule

```
crime_of_violence(a-object= @act1) ←
          hit(agent=X/person, object=Y/person) | @act1.
```

is interpreted as follows.

```
crime_of_violence(a-object=hit(agent=X/person, object=Y/person,
               place=W, time=Z)) ←
          hit(agent=X, object=Y, place=W, time=Z).
```

In addition to *H-term-variable*, *NHL* uses *H-term-constant* as H-term identifier. A *H-term-constant* is a symbol whose first character is "#." It also appears just after H-term. In H-term denoted by H-term-constant, omitted cases are treated as they have skolem constants even if the H-term appears in the condition part of a rule.

We call following form as an *extended H-term*.

H-term
H-term | *H-term-identifier*
H-term-identifier

(4) Unification

As well as ψ-terms, H-terms are unified by obtaining the greatest lower bound (GLB) of the type symbols in a subsumption relation that appear in the same position.

For example, consider following subsumption relation.

japanese_man $<_T$ japanese
japanese_man $<_T$ man
kill_with_blows $<_T$ strike
kill_with_blows $<_T$ kill

Then, the result of unifying the following H-terms

strike(agent=japanese)
kill(agent=man)

becomes the following H-term.

kill_with_blows(agent=japanese_man)

(5) Legal Rules

Legal rules take the following form.

> $u :: m_0 : h_0 \leftarrow m_1 : h_1, m_2 : h_2, ..., m_n : h_n.$
> (u is a unit name, m_i is a module name, h_i can be either *H-term* or *not H-term* whose cases can have *extended H-term*.)

Here, *unit name* is a rule name, and *not H-term* is an H-term to which "not" is attached.

While "$\neg$" indicates negation as conflict, "*not*" indicates negation as failure. There are three kinds of negation as failure (not_J, not_P, not_D).

Facts are represented by legal rules for which the right hand side is "true." We can thus omit the right-hand side to a represent fact.

(6) Case Rules

A Case rule takes the following form.

> $u :: m_0 : h_0 \leftarrow m_1 : h_1, m_2 : h_2, ..., m_n : h_n \mid CRC.$
> (u is a unit name, m_i is a module name, h_i is *H-term* or *not H-term* whose cases can have *extended H-term*.
> CRC is a constraint relaxation condition.)

NHL supports the control of unification of specific terms by describing the following conditions in CRC.

- "limit" specifies the upper limit of type generalization.
- "exact" indicates important types or terms. Important types or terms should always be satisfied, while other types or terms may not have.
- Function for measuring the rate of satisfied conditions on the right-hand side of a case rule.

As an example, consider the following case rule.

```
u:: liable(agent=tom, object=@injury) ←
      read_a_book(agent=jim, place=house1)|@reading,
      S/surprise(agent=tom, object=jim, place=house1)|@surprise,
      fall(agent=jim, source=sofa, place=house1)|@fall,
      T/causality(cause=@surprise, effect=@fall),
      injured(agent=jim, place=house1) |@injury,
      causality(cause= @fall, place= @injury),
      | { limit(surprise, surprise), limit(tom, person), limit(jim, person),
      limit(causality, causality), limit(injured, injured),
      exact(S), exact(T), exact(@injury), sim > 0.7 }.
```

The meaning of this rule is "while Jim was reading a book, Tom surprised him. Jim fell from the sofa, and was injured. In this case, Tom is liable for Jim's injury."

This rule is handled as follows.

(a) If A is a predicate or term in the rule, and if $limit(A, B)$ appears in CRC, A is replaced by B. If A is a predicate or term in the rule, and if $limit(A, B)$ doesn't appear in CRC, A is replaced by $\top$.

u:: liable(agent=X, object= @injury) ←
$\top$(agent=Y/person, place=Z/$\top$)|@reading,
S/surprise(agent=X/person, object=Y, place=Z)|@surprise,
$\top$(agent=Y, source=$\top$, place=Z)|@fall,
T/causality(cause= @surprise, effect= @fall),
injured(agent=Y, place=Z) |@injury,
causality(cause= @fall, effect= @injury),
| { exact(S), exact(T), exact(@injury), sim1 $>$ 0.7 }.

(b) If facts don't satisfy one of "important" types or terms, then this rule isn't fired.
(c) Otherwise, the satisfied conditions rate is measured. If the satisfied rate is more than a given constant, this rule will be fired. There are several functions which define the rates of satisfaction. Following is an example of such functions.

$$sim1 = \frac{P' * w1 + F' * w2}{P * w1 + F * w2}$$

(P and F are the number of types and terms in a case rule, and P' and F' are the number of satisfied types and terms. $w1$ and $w2$ are weights of importance).

Legal rules are classified into absolute rules and defeasible rules while all case rules are defeasible rules.

(7) Unit, standpoint, viewpoint

Even though a *unit* is defined as an identifier of a legal rule or a case rule, we can extend the definition of a unit to also represent a set of legal rules or case rules.

For example, we can define the unit *decision_of_supreme_court* whose members are u_i.

$$decision\_of\_supreme\_court :: \{u_1, u_2, ..., u_n\}$$

A *standpoint* is a set of priority relations between two units. For example, the standpoint "lex_superior" is that *decision_of_supreme_court* has priority over *decision_of_high_court*.

$$lex\_superior :: \{decision\_of\_high\_court <_S decision\_of\_supreme\_court\}$$

A *viewpoint* is a set of priority relations between standpoints. For example, the viewpoint "view1" is that *lex_superior* has priority over *lex_posterior*.

$$view1 :: \{lex\_posterior <_V lex\_superior\}.$$

(8) Argument

For a rule γ and a substitution θ, $\theta\gamma$ is an "*instance*" of γ. Here,

$$\theta = \{\phi_1/\psi_1, ..., \phi_n/\psi_n\}$$
(ϕ_i and ψ_i are ψ-term, and $\phi_i <_T \psi_i$)

Let Π be a set of rules. By eliminating *not H-term* from rules in Π, we can define Π'. "*Extension*" of Π ($Ext(\Pi)$)) is defined as a minimum set which satisfies follows.

- If $L_0 \leftarrow L_1, ..., L_m \in \Pi'$ and $L_1, ..., L_m \in Ext(\Pi)$ then $L_0 \in Ext(\Pi)$.
- For each pair p, $\neg q \in Ext(\Pi)$ and some substitution θ, if $\theta p = \theta q$, then $Ext(\Pi) = Lit$. Here Lit is a set of all possible *H-terms*.
- For each pair $r \in Ext(\Pi)$ and *not H-term* $\in \Pi$, for some substitution δ, if $\theta r = \theta H$, then $Ext(\Pi) = Lit$.

Let F be a set of all instances of absolute rules and let D be a set of defeasible rules. If and only if for some H-term p and any $D' \subset D$, $p \in Ext(F \cup D)$ and $p \notin Ext(F \cup D')$, then $A = (F, D)$ is an "*argument*" of p.

B directly defeats A, if B is a counter argument of A, if the top of the defeasible rule of B takes priority over the top of the defeasible rule of A, and if none of the subarguments of B is defeated by the others. $B defeats A$, if B directly defeats the subargument of A.

Arguments are classified into ***defeated arguments***, ***merely plausible arguments*** and ***justified arguments***. A defeated argument is one which is defeated by some other arguments. A merely plausible argument is one that is not defined by any counter argument. A justified argument defeats all counter arguments. A ***plausible argument*** is a merely plausible argument or a justified argument.

In the example of Figure 5, the argument "homicide" has an subargument "malice" in it. Though this subargument is directly defeated by an argument "negligence", its subargument "joke" is also directly defeated by an argument "serious." Therefore, "homicide" defeats "negligence", and if "negligence" is only one counter argument of "homicide", then "homicide" is justified argument.

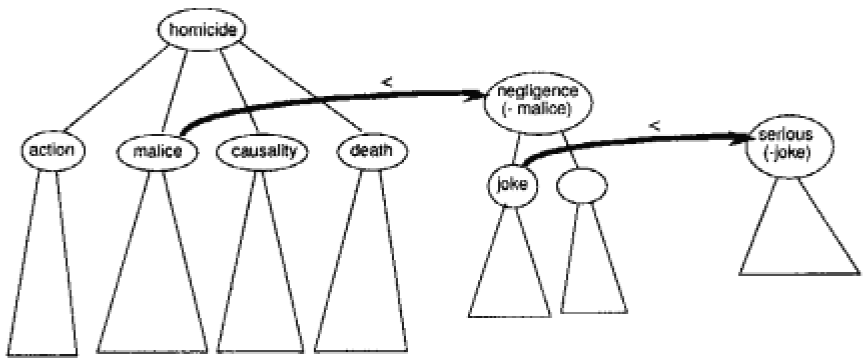


Figure 5: Justified argument

Let $A \vdash p$ mean that A is an argument for an H-term p. For a goal of a query "q", Ω is defined as $\{A \mid \exists\theta A \vdash \theta p\}$. If $A = (F, D) \in \Omega$ and $A' = (F, D') \in \Omega$ has a substitution θ which satisfies $\theta D = \theta D'$, $A \equiv A'$ holds and $\Omega / \equiv = \{\omega_1, ..., \omega_n\}$ is defined. MGA (Most General Argument) of ω_i is an argument $A_i = (F, D_i) \in \omega_i$ and $\forall A_{i'} = (F, D_{i'}) \in \omega_i$, there exists a substitution θ which satisfies $D_{i'} = \theta D_i$. The "correct answer argument" is a set of MGA in $\Omega / \equiv$.

The query of NHL consists of a goal q, facts F, a viewpoint V and control parameters C as follows.

?- $solve(q, F, V, C, Ans)$.

Following is an example of a query.

?- $solve(satisfy(crime\_of\_homicide(obj = shot\_a\_gun(agent = tom, object = jim)),$
$case01, viewpoint02, [analysis\_mode, threshold(60)], Ans).$

Ans contains all plausible arguments in "correct answer argument" in the form of an argument table.

An *argument table* contains the arguments that will produce the given goal, pointers to possible counter arguments, relative strength comparing to counter arguments and category of each subargument (Fig.6).

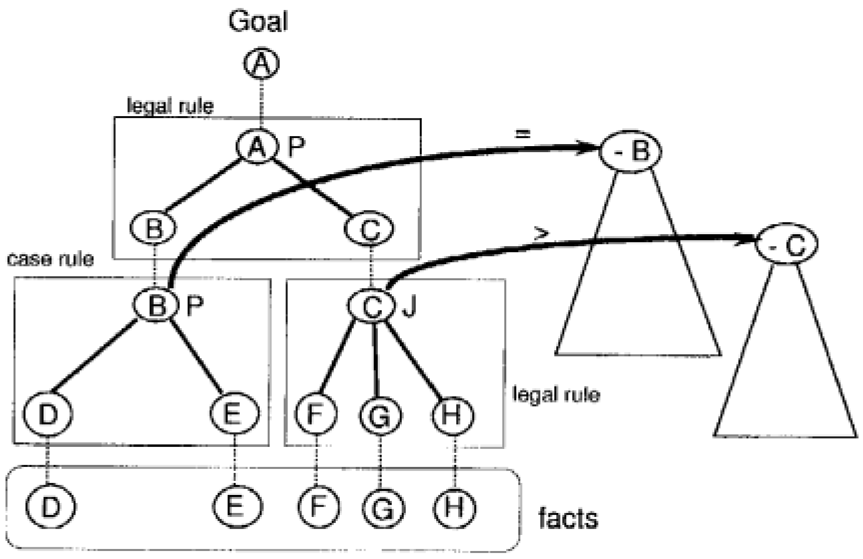


Figure 6: Argument table

(9) Meta rules

There are two types of meta rules.

1. Interpretation of Legal Rules:

   $meta1 :: expand(LegalRule, Control, \wedge ModifiedLegalRules)$
   $meta2 :: reduce(LegalRule, Control, \wedge ModifiedLegalRules)$

   This meta rule generalize or specialize an legal rule.

2. Modification of initial facts:

   $meta3 :: modify(Facts, Control, \wedge newFacts)$

   This meta rule asks the user to modify the initial facts.

Priorities between rules are also defined on these meta rules. Therefore, a viewpoint has information of priority between legal rules, case rules, interpretation of legal rules and modification of initial facts (evidence).

(10) Representation of Legal Knowledge

Legal reasoning uses various kind of legal knowledge. They are represented as legal rules, case rules, type or term, meta rules and priority between rules (Fig. 7).

Statute	legal rule
Theory	legal rule
Old Case (Induced rule)	legal rule
Old Case (Judgement)	case rule
Concept	type / term
Commonsense	legal rule
Interpretation	meta rule
Standpoint / Viewpoint	priority

Figure 7: Representation of legal knowledge

4 Example

Using an example, we will explain how arguments are obtained in *NHL*.

Case: "Tom caused a traffic accident while he was driving a car, and Jim was injured. Tom got off a car and watched Jim. Tom mistook as Jim was dead, and he ran away leaving Jim there. Jim was frozen and died. Should Tom be punished by the crime of aggravated desertion resulting in death ?" (Fig. 8).

This example seems to be easy hit-and-run case. However, concerning the crime of aggravated desertion resulting in death, there are four issue points as follows.

(1) Does Tom has obligation to protect Jim?

- (1p) When a traffic accident occurred, the driver has obligation to protect the sufferer.
- (1d) The obligation to protect the sufferer in article 218 of the Penal Code should be interpreted strictly. Tom didn't take care of Jim because he mistook Jim was dead. Therefore, Tom doesn't have obligation to protect Jim.

(2) Did Tom desert Jim ?

- (2p) Leaving a person alone corresponds to desertion even if the victim was left in the safe place.
- (2d) Desertion should be interpreted to take a person to dangerous place. Therefore, Tom's action is not desertion.

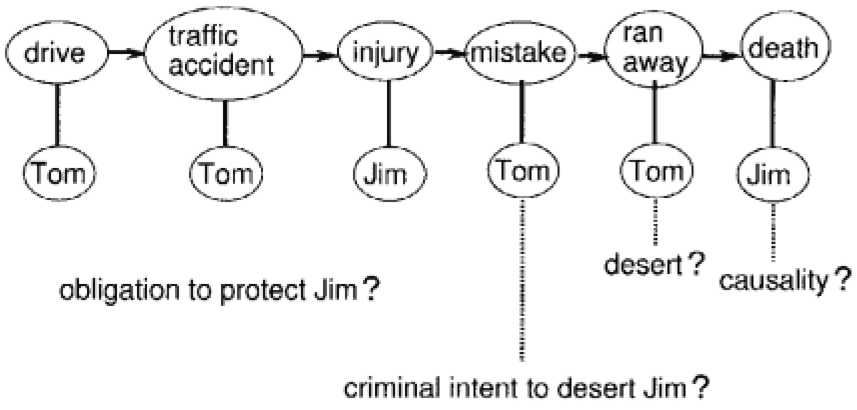


Figure 8: Example Case

(3) Did Tom has criminal intent to desert Jim ?

(3p) Though Tom mistook Jim was dead and he didn't have intent to desert alive person, he has a criminal intent to desert Jim because such action should be to blame. (Loose Intent Theory)

(3d) As the purposes of litigation of crime of abandon of corps and crime of desertion, Tom doesn't have a criminal intent to desert Jim. (Purpose Intent Theory)

(3p') Even if the purposes of litigation of crime of abandon of corps and crime of desertion, Tom has a criminal intent to desert Jim because abandonment of corps and desertion of a person has the common features. (Same rule was used in an old case of importing stimulants.)

(3d') As Tom doesn't have intent to desert Jim, he doesn't have criminal intent at all. (Strict Intent Theory)

(4) Is there causality between desertion and Jim's death.

(4p) If Tom hadn't deserted Jim, Jim didn't die. Therefore, thereis a causality between desertion and death. (Condition Causality Theory)

(4d) As desertion is the main reason of Jim's death, there is causality. (Main Reason Causality Theory)

Concerning these issue points, different lawyers have different viewpoints and support different theories or old cases.

For example, if we focus on the public interests, we put importance on psychological aspect of Tom, and select the loose intent theory. On the contrary, if we focus on the personal interests, we put importance on the result of Tom's action, and select the strict intent theory.

Like this, the plausible argument depends on the viewpoint. By preparing typical patterns of viewpoints beforehand and by selecting one of them, we can draw different arguments.

5 Conclusion

We presented three layers model of legal reasoning and a knowledge representation language NHL.

This three layer model covers wide range of legal reasoning, and it is possible to control the inference using various rules such as legal rule, case rule and meta rules by using viewpoint.

NHL is has several mechanisms to develop a legal reasoning system in it. Except meta rules, we have been developing the first version of NHL on logic programming language KLIC. As KLIC programs run on the Unix environment, NHL has portability. New HELIC-II system is developed using NHL.

Other reseach subjects of the new HELIC-II are a conceptional dictionary, analysis of viewpoint in the criminal law, graphic user interface and a debate model which belongs to the debate layer in figure 1. We will finish the development of the new HELIC-II by next March. Though the current target domain of the new HELIC-II is the criminal law, it is applicable to other domain because its framework is general.

References

[Ait-Kaci 86] Ait-Kaci, H. and Nasr,R. *"LOGIN: A Logic Programming Language with Built-in Inheritance."* in *"J. Logic Prog."*,1986, vol.3, pp. 185-215

[Branting 89] L.K.Branting : *"Representing and Reusing Explanations of Legal Precedents"* in *"International Conference on Artificial Intelligence and Law "*, 1989.

[Chen 89] Chen,W.,Kifer,M.and Warren,D.S. *"HiLog as a Platform for Database Language"* in *"2nd Intl. Workshop on Database Programming Languages"*, Oregon Coast,OR,June 1989.

[Chikayama 89] Chikayama, T,ICOT. *"A Portable and Reasonably Efficient Implementation of KL1."* in *"International Conference on Logic Programming"*, The MIT Press,1993.

[Gelfond 90] Gelfond,M.,and V.Lifschitz. *"Logical Programs with Classical Negation."* in *"Proceedings of the Seventh International Logic Programming Conference"*, New York:MIT Press, 1990, pp.579 597.

[Nitta 92] Nitta, K. *"HELIC-II: a legal reasoning system on the parallel inference machine."* in *"Proceedings of FGCS92"*, 1992, pp.1115-1124.

[Prakken 93] Prakken,H. *"A logical framework for modeling legal argument"* in *"The Fifth International Conference on Artificial Intelligence and Law"*,1993,pp.1-9

[Smolka 89] Smolka,G.and Ait-Kaci.H. *"Inheritance Hierarchies:Semantics and Unification"* in *"J. Symbolic Computation"*,1989, pp.343 379

[Sartor 93] Sartor,G. *"A Simple Computational Model for Nonmonotonic and Adversarial Legal Reasoning"* in *"The Fifth International Conference on Artificial Intelligence and Law"*,1993,pp.192-201

[Taki 92] Taki,K. *"Parallel Inference Machine PIM."* in *"Proceedings of FGCS92"*, 1992, pp.50-72.