PROCEEDINGS OF THE INTERNATIONAL CONFERENCE
ON FIFTH GENERATION COMPUTER SYSTEMS 1992,
edited by ICOT.

HELIC-II:
A Legal Reasoning System on the Parallel Inference Machine

Katsumi Nitta (1) Yoshihisa Ohtake (1) Shigeru Maeda (1)
Masayuki Ono (1) Hiroshi Ohsaki (2) Kiyokazu Sakane (3)

(1) Institute for New Generation Computer Technology
4-28, Mita 1-chome, Minato-ku, Tokyo 108, Japan
(2) Japan Information Processing Development Center
(3) Nippon Steel Corporation

nitta@icot.or.jp

Abstract

This paper presents HELIC-II, a legal reasoning system on the parallel inference machine. HELIC-II draws legal conclusions for a given case by referring to a statutory law (legal rules) and judicial precedents (old cases). This system consists of two inference engines. The rule-based engine draws legal consequences logically by using legal rules. The case-based engine generates legal concepts by referencing similar old cases. These engines complementally draw all possible conclusions, and output them in the form of inference trees. Users can use these trees as material to construct arguments in a legal suit.

HELIC-II is implemented on the parallel inference machine, and it can draw conclusions quickly by parallel inference.

As an example, a legal inference system for the Penal Code is introduced, and the effectiveness of the legal reasoning and parallel inference model is shown.

1 Introduction

The primary knowledge source of a legal inference system is a statutory law. A statutory law is a set of legal rules. As legal rules are given as logical sentences, they are easily represented as logical formulae. Therefore, if a new case is described using the same predicates as those appearing in legal rules, we can draw legal conclusions by deductive reasoning.

However, legal rules often contain legal predicates (legal concepts) such as "public welfare" and "in good faith". Some legal concepts are ambiguous and their strict meanings are not fixed until the rules are applied to actual facts. Predicates which are used to represent actual facts do not contain such legal concepts. As there are no rules to define sufficient conditions for legal predicates, in order to apply legal rules to actual facts, interpreting rules and matching between legal concepts and facts are needed. To realize this, precedents (old cases) are often referenced because they contain the arguments of both sides (plaintiff vs. defendant or prosecutor vs. defendant) and the judge's opinions concerning interpretation and matching.

Consequently, legal reasoning can be modeled as a combination of logical inference using legal rules and case-based reasoning using old cases. Based on this model, several hybrid legal inference systems consisting of two inference engines have been developed [Rissland *et al.* 1989] [Sanders 1991(a)]. However, as practical legal systems contain many legal rules and old cases, it takes a long time to draw conclusions. Moreover, controlling two engines often requires a complex mechanism.

ICOT (Institute for New Generation Computer Technology) has developed parallel inference machines (Multi PSI and PIMs) [Uchida *et al.* 1988],[Goto *et al.* 1988]. These are MIMD-type computers, and user's programs written in parallel logic programming language KL1 [Chikayama *et al.* 1988] are executed in parallel on them.

The **HELIC-II** (Hypothetical Explanation constructor by Legal Inference with Cases by 2 inference engines) is a legal inference system based on the hybrid model. It has been developed on the parallel inference machine, and draws legal conclusions for a given case by quickly referencing statutory law and old cases.

In Section Two, we introduce the function and architecture of HELIC-II. In Section Three, we explain legal knowledge representation. In Section Four, we explain the reasoning mechanism of HELIC-II. In Section Five, a legal inference system of the Penal Code is explained.

2 Overview of HELIC-II

The function of HELIC-II is to generate all possible legal conclusions for a given case by referring to legal rules

and old cases. These conclusions are represented in the form of inference trees which include final conclusions and explanations of them.

HELIC-II consists of two inference engines - the rule based engine and the case-based engine - and three knowledge sources - a rule base, a case base and a dictionary of concepts (see Fig.1). The rule-based engine refers to legal rules and draws legal consequences logically. The case-based engine generates abstract predicates (legal concepts) from concrete predicates (given facts) by referring to similar old cases.

HELIC-II draws legal consequences using these two engines. Since the reasoning of these engines is data-driven, there are no special control mechanisms to manage this. A typical pattern of reasoning by HELIC-II is as follows. When a new case (original facts) is given to HELIC-II, the case-based engine initially searches for similar old cases and generates legal concepts which may hold in the new case. These concepts are passed to the rule-based engine by way of working memory(WM). Then, the rule-based engine draws legal consequences using original facts and legal concepts.

These results are gathered by an explanation constructor, which then produces inference trees.

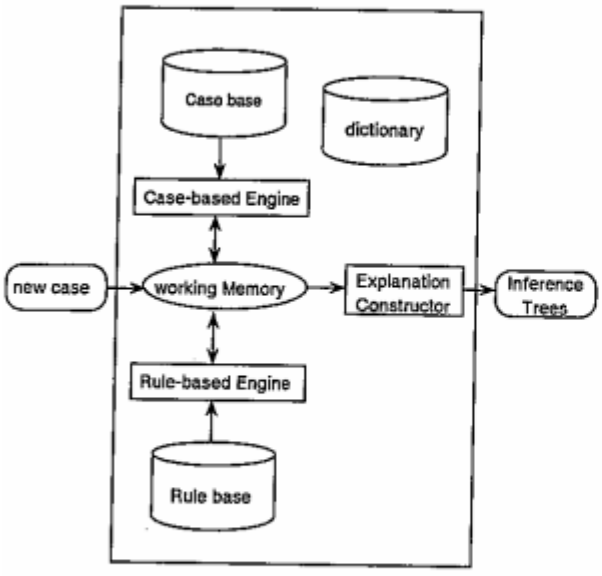

Figure 1: The architecture of HELIC-II

3 Knowledge Representation

In this section, we will explain the representation of legal knowledge in HELIC-II. We will show how to represent legal rules, old cases and legal concepts.

3.1 Representation of Legal Rules

A statutory law consists of legal rules. Each legal rule is represented as follows.

$$RuleName(Comment, RuleInfo, [A_1, A_2, ..., A_i] \rightarrow [[B_1, .., B_k], [C_1, .., C_l], ..]).$$

In this clause, *RuleName* is the rule identification, *Comment* is a comment for users and *RuleInfo* is additional information such as article number. The LHS ($[A_1, A_2, ..., A_i]$) is the condition part, and the RHS($[[B_1, .., B_k], [C_1, .., C_l], ..]$) is the consequence part. $[B_1, .., B_k]$ and $[C_1, .., C_l]$ are combined disjunctively. Each literal of the LHS and RHS is an *extended predicate* or its negation (denoted by "$\sim$" or "*not*"). An *extended predicate* consists of a predicate (concept), an object identifier and a list of *attribute* = *value* pairs. The following is an example of an extended predicate. An object "*drive1*" is an instance of a concept "*drive*". Two *attribute* = *value* pairs ($agent = tom$ and $car = toyota1$) are defined.

$$drive(drive1, [agent = tom, car = toyota1]).$$

Internally, this extended predicate is treated as a set of the triplet $\{object, attribute, value\}$ as follows.

$$\{drive1, agent, tom\}$$
$$\{drive1, car, toyota1\}$$

In a clause, we can use "*not*" (negation as failure) in addition to "$\sim$" (logical not). By introducing "*not*", nonmonotonic reasoning is realized, and the representation of exceptional rules and presumed facts are easily represented [Sartor 1991].

The following are examples of legal rules.

$$homicide01(\text{"}example\text{"}, [article = 199],$$
$$[person(A), person(B),$$
$$action(Action, [agent = A]),$$
$$intention(Intention, [agent = A, action = Action,$$
$$goal = Result]),$$
$$death(Result, [agent = B]),$$
$$caused(Caused, [event = Action, effect = Result2]),$$
$$death(Result2, [agent = B]),$$
$$not(\sim illegality(Illegal, [agent = A,$$

$action = Action, result = Result2]))]$
$\rightarrow$
$[[crimeOfHomicide(Crime, [agent = A,$
$action = Action, result = Result2])]]).$

$legality01(\text{"}example\text{"}, [article = 38],$
$[action(Action, [agent = A]),$
$intention(Intention, [agent = A, action = Action,$
$goal = Result]),$
$selfDefence(Result, [object = Action]),$
$caused(Caused, [event = Action, effect = Result2])]$
$\rightarrow$
$[[\sim illegality(Illegal, [agent = A,$
$action = Action, result = Result])]]).$

The first rule is a definition of the crime of homicide, which is given by the Penal Code.

The meaning of "$not(\sim illegality(Illegal, [...]))$" is that illegality is presumed, in other words, if there isn't proof that "$\sim illegality(Illegal, [...])$" holds then "$not(\sim illegality(Illegal, [...]))$" is true.

The second rule is an exception of the first rule. If a person did some action in defense, "$illegality(Illegal, [...])$" is refuted.

3.2 Representation of Cases

A judicial precedent consists of the arguments of both sides, the opinion of the judges and a final conclusion. We represent a precedent (an old case) as a *situation* and some *case rules*, and represent a new case as a *situation*.

(1) Situation

A *situation* consists of a set of events/objects and their *temporal relations*. An event and an object are represented as an extended predicate as introduced in the previous section. The temporal relations are represented as follows.

$problem(CaseID, Comment, TemporalRelations).$

CaseID is the case identification, *Comment* is a comment for users and *TemporalRelations* is a list of relations between events. To represent temporal relations between events/objects, we use Allen's interval notation such as "$before$", "$meets$", "$starts$", and so on [Allen 1984].

The following is an example of a situation.

$problem(trafficAccident112, \text{"}example\text{"},$
$[before(dinner1, drive1), during(accident1, drive1)]).$
$dinner(dinner1, [agent = john, place = maxim's]).$
$drive(drive1, [agent = john, car = toyota1]).$
$accident(accident1, [agent = john]).$
$person(john, [sex = male]).$
$person(mary, [sex = female]).$
$restaurant(maxim's, [rank = 5stars]).$
$car(toyota1, [type = sportsCar]).$

The meaning of this example is that the case "$traffic\ accident\ 112$" consists of three events such as "$dinner1$", "$drive1$" and "$accident1$". "$Dinner1$" occurred before "$drive1$", and "$accident1$" happened during "$drive1$". The event "$dinner1$" is a lower concept of "$dinner$", and it is acted by "$john$" in "$maxim's$", etc..

(2) Case Rules

Arguments by both sides are represented as a set of *case rules*. The following is the syntax of a case rule.

$RuleName(Comment, RuleInfo,$
$[A_1, A_2, ..., A_i] \rightarrow [B_1, B_2, .., B_k]).$

RuleName is the rule identification, *Comment* is a comment for users and *RuleInfo* is additional information such as a related article, index for the opposing side's case rules, relation to judge's decision and so on. The LHS ($[A_1, A_2, ..., A_i]$) is the context of the opinion, and the RHS ($[B_1, B_2, .., B_k]$) is the conclusion insisted on by one side.

The following is an example of a case rule.

$rule001(\text{"}example\text{"},$
$[\ article = 218, insisted = prosecutor,$
$result = lost],$
$[\ drive(drive1, [agent = john/important,$
$object = toyota1/trivial]),$
$person(john, [sex = male/trivial]),$
$person(mary, [sex = female/trivial]),$
$accident(accident1, [agent = john/important]),$
$caused(caused1, [event = accident1/important,$
$effect = injury1/important]),$
$injury(injury1, [agent = mary/trivial])]$
$\rightarrow$
$[\ responsibility(resp1, [agent = john,$
$object = ken, reason = accident1])]).$

The meaning of this case rule is:"In the case that a traffic accident caused by John injured Mary, John had a responsibility of care to Mary." This rule concerns article 218 of the Penal Code and was insisted on by the prosecutor, but the judge didn't employ this rule. On the LHS, "$effect = injury1$" is an important fact from

the legal point of view. Therefore, this fact is marked as "important". We can use "exact", "important" and "trivial" to represent levels of importance. This information is used to calculate the similarity between two situations.

Arguments in a case are sequences of case rules. As both sides try to draw contradictory conclusions, an old case contains case rules whose conclusions are inconsistent.

3.3 Representation of Concepts

All concepts in legal rules and cases must be contained in the dictionary. In other words, each event and object in a situation are instances of these concepts.

In the dictionary, a *super concept*, a *concept* and a list of *attributes* are defined as follows.

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

The similarity between concepts is defined by the *distance* in the hierarchy (see Fig.2). For example, "*baby*" is closer to "$infant$" than to "*lion*" because it requires two steps for "*baby*" to reach "$infant$" but three steps to reach "*lion*" in this hierarchy.

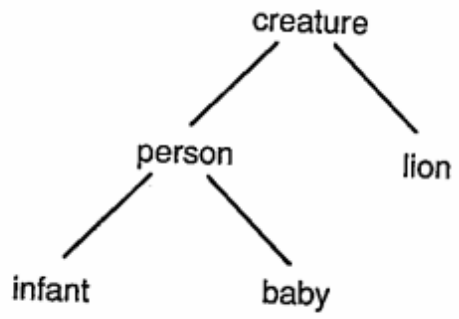

Figure 2: Hierarchy of concepts

4 Reasoning by HELIC-II

In this section, we will explain the reasoning mechanisms of the rule-based engine and the case-based engine. These engines are implemented in the parallel logic programming language KL1 and run on the parallel inference machine.

4.1 A Rule-based Engine

The function of the rule-based engine is to draw all legal consequences by the forward reasoning of legal rules, using original data (a new case) and results from a case-based engine.

The rule-based engine is based on the parallel theorem prover MGTP (Model Generation Theorem Prover)[Fujita *et al.* 1991] developed by ICOT.

MGTP solves range restricted non-Horn problems by generating models. For example, let's take the following clauses.

C1: $true \rightarrow p(a); q(b)$.
C2: $p(X) \rightarrow q(X); r(X)$.
C3: $r(X) \rightarrow s(X)$.
C4: $q(X) \rightarrow false$.

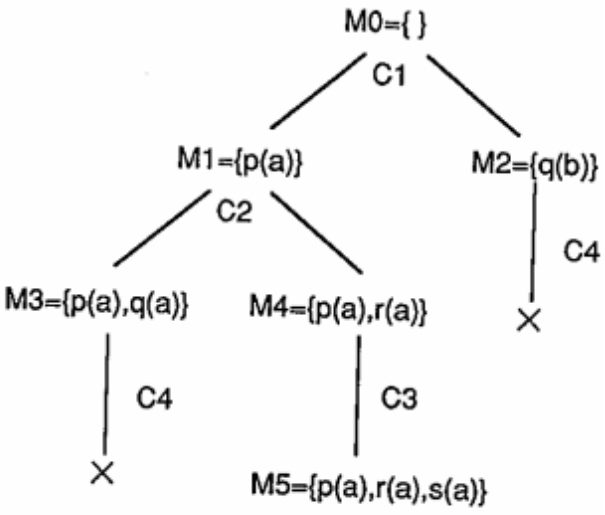

Figure 3: MGTP proof tree

MGTP calculates models which satisfy these clauses as follows (see Fig.3). The proof starts with null model $M0 = \{\phi\}$. By applying C1, $M0$ is extended into $M1 = \{p(a)\}$ and $M2 = \{q(b)\}$. Then, by applying C2, $M1$ is extended into $M3 = \{p(a), q(a)\}$ and $M4 = \{p(a), r(a)\}$. Using C4, $M3$ and $M2$ are discarded. By C3, $M4$ is extended to $M5 = \{p(a), r(a), s(a)\}$. $M5$ is a model which satisfies all clauses.

In MGTP, each clause is compiled into a KL1 clause, and each KL1 clause is applied in parallel on the parallel inference machine. In the problem in which the proof tree has many branches, parallel inference performance becomes high.

To use MGTP as a rule-based engine of HELIC-II, we extended the original MGTP as follows.

1. **Realization of "not (negation as failure)":** We made MGTP able to treat "negation as failure" based on [Inoue *et al.* 1991]. For example, the following C is treated as C', and the model is extended in two ways (see Fig.4). Here, "k" is a modal operator, and "k(r(X))" means that the model is believed to contain a datum which will satisfy r(X) in the future.

   C: $not(r(X)) \rightarrow s(X)$.
   C': $dom(X) \rightarrow k(r(X)); \sim k(r(X)), s(X)$.

   After MGTP generates models which satisfy all clauses, the rule-based engine examines each of them. For example, if a model contains both $\sim k(r(a))$ and $r(a)$, or if a model contains $k(r(a))$ and doesn't contain $r(a)$, the model is discarded.

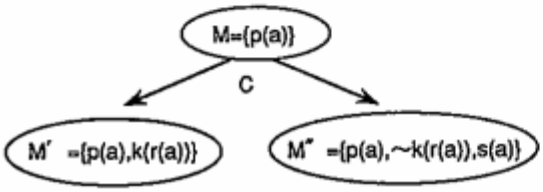

Figure 4: Negation as failure of MGTP

2. **Realization of the multiple context:** The rule-based engine uses both original facts (a new case) and results from the case-based engine as the initial model. The case-based engine may generate data which conflicts with each other such as "$q(b)$" and "$\sim q(b)$". Therefore, before reasoning, the rule-based engine has to split the initial model into several ones so that each model doesn't contain any conflicts (see Fig. 5).

   However, the case-based engine has not generated all results when the rule-based engine begins to reason because the reasoning of both engines is data driven. To obtain the pipeline effect, we developed a function to register predicates which may cause conflicts, and to split the model when such predicates reach the rule-based engine. For example, in Fig.5, if $\sim q(b)$ reaches the rule-based engine, the model is split before $q(b)$ is reached. We implemented this mechanism by using a similar modal operator as the "k-operator".

3. **Keeping justification:** To construct inference trees, the rule-based engine must keep the justifications for each consequence. A justification consists of a rule name and data which matches the LHS of the rule.

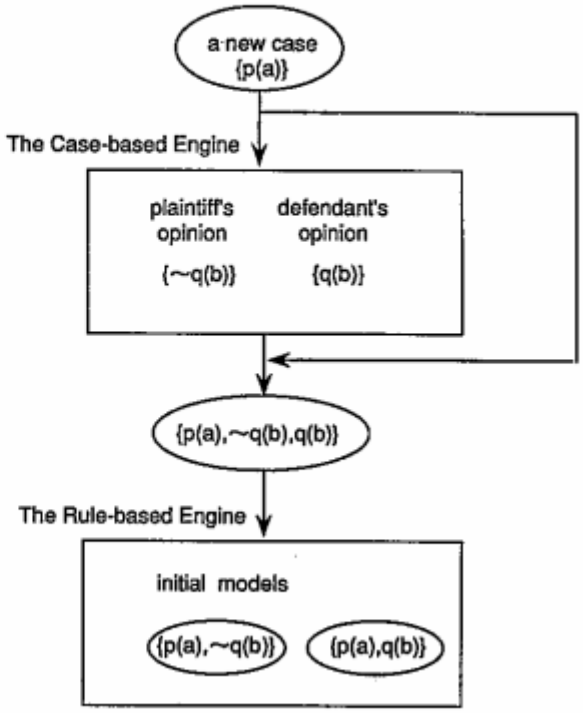

Figure 5: Splitting a model

4. **Temporal reasoning:** We prepared a small rule set of temporal reasoning [Allen 1984] to help in describing the temporal relation. The following are example rules.

   $before(A,B), before(B,C) \rightarrow before(A,C)$.
   $meets(A,B), overlaps(C,B) \rightarrow$
   $overlaps(A,C); during(A,C); starts(A,C)$.

With these extensions, the rule-based engine has many proof tree branches even if clauses don't have the disjunction such as C1 and C2 in Fig.3. Therefore, the rule-based engine has a lot of parallelisms in its reasoning.

4.2 A Case-based Engine

The function of the case-based engine is to generate legal concepts by using similar old cases. The reasoning of the case-based engine consists of two stages (see Fig.6).

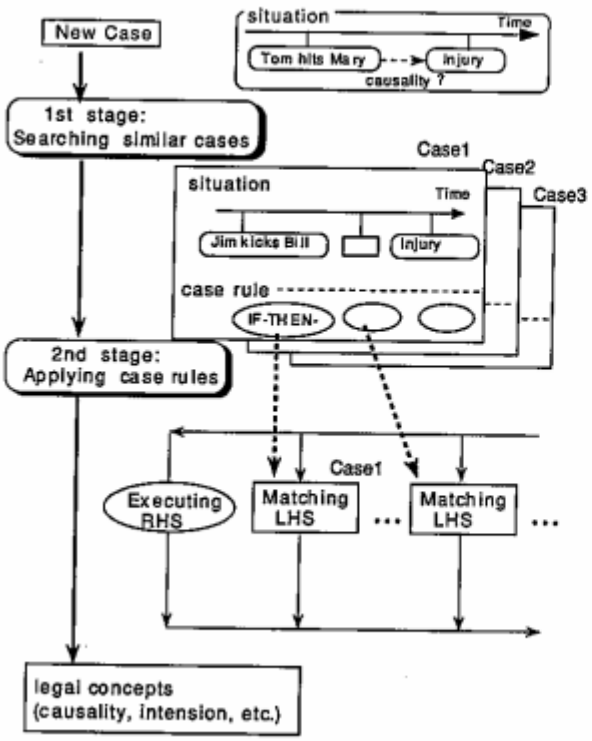

Figure 6: Reasoning by the case-based engine

1. **Searching similar cases:**

   The role of the first stage is to search for similar cases from the case base. At first, the case-based engine constructs a *sequence of events* for each case. As the situations of the new case and old cases are described as a set of events/objects and their temporal relations, it is easy to construct a *sequence of events* for each situation.

   Then, the case-based engine tries to extract common subsequences from event sequences of the new case and each old case. For example, let's take the following two sequences.

   $$\begin{aligned} \text{S1: } & [..., meets(strike1, injury1), \\ & \quad during(runAway1, injury1), ..] \\ \text{S2: } & [..., before(kick2, sneak2), ..] \end{aligned}$$

   In this example, the temporal relation between "strike1" and "runAway1" is the same as that of "kick2" and "sneak2". Furthermore, "strike1" and "kick2" have a common upper concept "violence", and "runAway1" and "sneak2" have a common upper concept "escape" in the dictionary. Therefore, we regard [strike1, runAway1] and [kick2, sneak2] as mapped subsequences of S1 and S2 (see Fig.7).

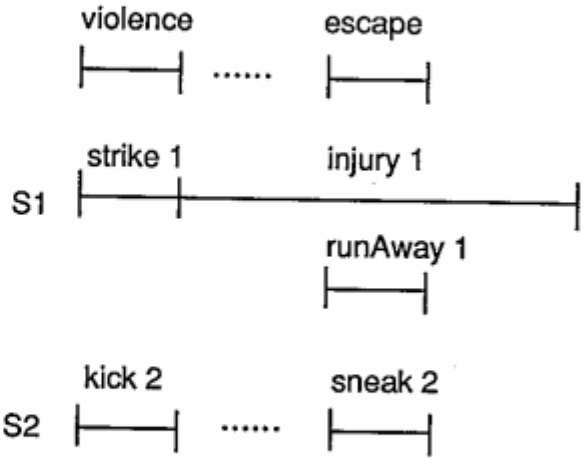

Figure 7: Subsequence of events

   The similarity between two cases is evaluated by the length of the longest mapped subsequence. Several cases whose similarities are beyond a threshold are selected in the first stage.

2. **Applying case rules:**

   The role of the second stage is to apply the case rules of selected cases as follows [Branting 1989].

   At first, the similarity between the LHS of a case rule and a new case is evaluated. For example, let's take "rule001" in section 3.2 and the following new case.

   $$\begin{aligned} & person(bill, []). \\ & baby(jane, []). \\ & cycle(cycle2, [agent = bill, object = honda2]). \\ & collision(collision2, [agent = bill]). \\ & sprain(sprain2, [agent = jane]). \\ & intention(intention2, [goal = injury2]). \\ & injury(injury2, [agent = jane]). \end{aligned}$$

   The engine tries to map the LHS of "rule001" to a new case. As the following pairs of event/object have common upper concepts in the dictionary, we map these pairs (see Fig.8).

john	$\leftrightarrow$	bill
mary	$\leftrightarrow$	jane
drive1	$\leftrightarrow$	cycle2
toyota1	$\leftrightarrow$	honda2

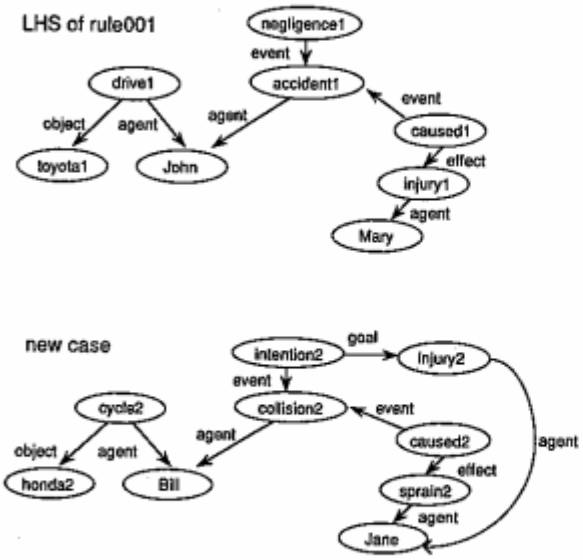

Figure 8: Mapping networks

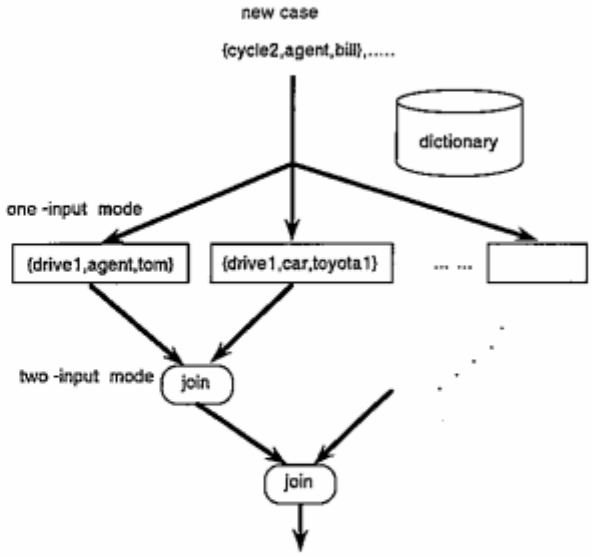

Figure 9: Rete-like networks of KL1 processes

accident1	↔	collision2
injury1	↔	sprain2
caused1	↔	caused2

The similarity is evaluated by counting the number of mapped links in Fig.8. As we explained in section 3.2, an annotation (exact, important, trivial) is attached to each link in the network. These annotations and the distances between concepts are used as weights to evaluate similarities. Even if some conditions of a case rule are not satisfied, but the important conditions are satisfied, then the LHS may be judged as similar to the new case. For example, in Fig.8, though there is no node which can be mapped to "negligence1", "rule001" may be selected as similar.

Next, the case-based engine selects case rules whose LHSes are similar to the new case, and executes their RHSes.

The matching and executing case rules are repeated until there are no case rules left to be fired.

On the parallel inference machine, each stage is executed in parallel. In the first stage, before searching, cases are distributed to processors (PEs) of the parallel inference machine, and then a new case is sent to each PE. Each PE evaluates similarities between the new case and old cases, and selects similar ones.

In the second stage, case rules are distributed to PEs, and the LHSes of each case rule are compiled into a Rete-like network of KL1 processes (see Fig.9). Then, triplets $(\{object, attribute, value\})$ which are facts of the new case are distributed to each PE as tokens. To realize matching based on similarity, each one-input node refers to the dictionary of concepts, and each two-input node not only examines the consistency of pairs of tokens but evaluates their similarities with the LHS.

5 A legal reasoning system for the Penal Code

We developed an experimental legal reasoning system for the Penal Code.

In the Penal Code, general provisions and definitions of crimes are given as legal rules. Though they seem to be strictly defined, the existence of *criminal intention* and *causality* between one's action and its result often becomes the most difficult issue in the court. The concept of *causality* in the legal domain is similar to the concept of *responsibility* and is different from physical causality. Therefore, to judge the existence of causality, we have to take into account various things such as social, political and medical aspect.

We show the function of the reasoning system of the Penal Code using Mary's case. We selected this case

from the qualification examination for 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 the police station. However, Jim caused an accident on the way to the police. 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. The hard issues of this case are the following.

1. **Causality between Mary's action and Tom's death:**

   If Mary hadn't abandoned Tom, Tom wouldn't have died. Moreover, the reason for his death wasn't injury but freezing. Therefore, some lawyers will judge the existence of causality and insist she should be punished for the crime of "*abandonment by person responsible resulting in death*". On the other hand, other lawyers will deny any causality because causality was interrupted by Jim's action.

2. **Causality between Jim's action and Tom's death:**

   Jim did several actions such as "pick up", "drive", "cause accident" and "leave Tom". Among them, "cause accident" will be punished by the crime of "*injury by negligence in the performance of work*", and "leave Tom" will be punished by the crime of "*death by negligence*". Moreover, if there is causality between "cause accident" and Tom's death, Jim will be punished by the crime of "death by negligence in the performance of work" which is very grave. As the main reason of Tom's death is freezing, it is difficult to judge the causality.

Though the Penal Code has no definite rule for the causality, lawyers can get hints from old cases. For example, let's take Jane's case which was handled by the Supreme Court in Japan.

Jane's Case:

Jane strangled Dick to kill him. Though Dick only lost consciousness, Jane thought he was dead. Then, she took him to the seashore, and left him there. He inhaled sand and suffocated to death.

In the court, there were arguments between the prosecutor and Jane. The prosecutor insisted Jane should be punished by the crime of *homicide* because of the following reasons.

P1: "Strangling" and "taking to the seashore" should be considered the one action of performing the homicide. Therefore, it is evident that there was an intention to kill Dick and causality between her action and Dick's death.

P2: There is causality between "strangling" and "Dick's death" even though "strangling" wasn't the main reason for his death.

On the contrary, Jane insisted her actions didn't satisfy the condition of the crime of homicide because of the following reason.

J1: "Strangling" should be punished be the crime of "*attempted homicide*, and "taking to the seashore" should be punished by the crime of "*manslaughter caused by negligence*" because there isn't causality between strangling and Dick's death, and there wasn't an intention to kill him when taking him to the seashore.

We represent Mary's situation and Jane's case rule as follows.

Mary's situation

$problem(\text{"}mary's\ case\text{"}, \text{"}example\text{"}, ...).$
$abandon(aba1, [agent = mary, object = tom]).$
$pickup(pic2, [agent = jim, object = tom]).$
.....
$traffic Accident(acc1, [agent = jim]).$
.....

Jane's opinion

$rule002(\text{"}Jane's\ case\text{"},$
$[article = 218, insisted = defendant,$
$result = lost],$
$[\ suffocate(suf1, [agent = jane/trivial,$
$object = dick/trivial]),$
$intention(int1, [agent = jane/trivial,$
$object = act1/important,$
$goal = death1/important]),$
$death(death1, [agent = dick/trivial]),$

$caused(caused1, [event = act1/important,$
$effect = lost1/important]),$
.....
$\rightarrow$
$[\ \sim caused(caused1, [event = act1,$
$effect = death3])]).$

The case-based engine of HELIC-II generated "$\sim caused(ID, [event = acc1, effect = death9])$" by applying *rule002*.

In Mary's case, HELIC-II generated 12 inference trees. Some of them are based on the prosecutor's opinion and others are based on the defendant's opinion. The root of each tree is a possible crime such as *abandonment by a person responsible resulting in death, manslaughter caused by negligence*, etc.. The leaves are the initial data of the new case, and intermediate nodes are consequences by case rules or legal rules (see Fig.10).

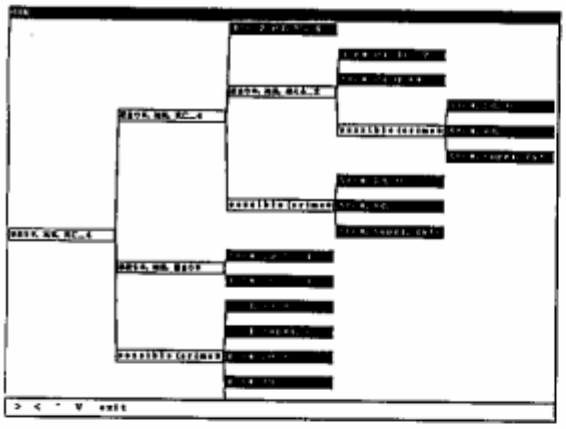

Figure 10: An Inference Tree

We measured the calculation time to draw a conclusion for Mary's case on the experimental parallel inference machine Multi-PSI. The number of rules used was about 20 and the number of cases used was about 30.

Fig.s 11 and 12 show the performance of the case-based engine, and Fig.13 shows the performance of the rule-based engine. These graphs show the effectiveness of the parallel inference.

6 Conclusion

We introduced the parallel legal reasoning system HELIC-II. The advantages of HELIC-II are as follows.

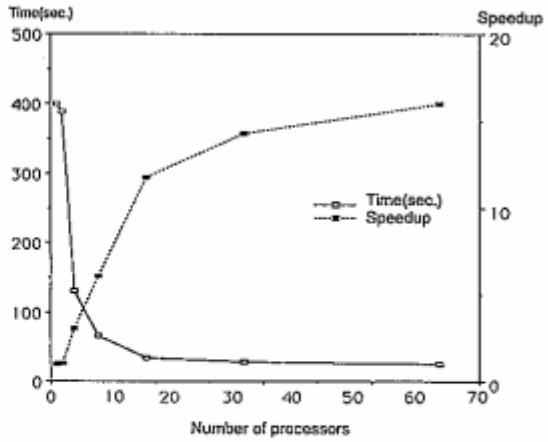

Figure 11: Performance of stage 1 of the case-based engine

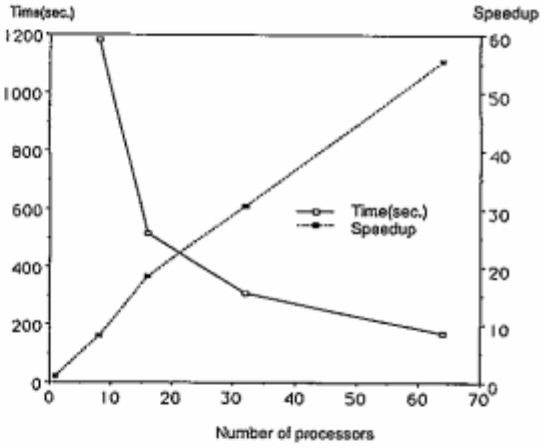

Figure 12: Performance of stage 2 of the case-based engine

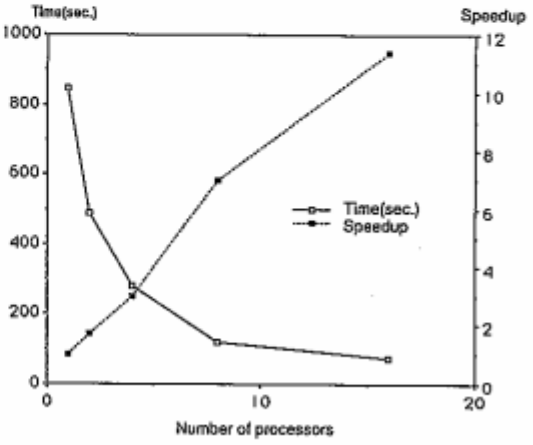

Figure 13: Performance of the rule-based engine

1. The hybrid architecture of HELIC-II is appropriate to realize legal reasoning. As the reasoning of both engines is data-driven, controlling these engines is easier.

2. The knowledge representation and inference mechanisms of HELIC-II are simple but convenient to represent legal rules and old cases.

3. By parallel inference, HELIC-II draws conclusion quickly. As the rule base and the case base of the legal domain are very large, quick searching and quick reasoning are important to develop practical systems.

4. Though it is troublesome to represent cases in detail, the rules of temporal reasoning help to describe cases.

There are many tasks for extending HELIC-II. The following are examples.

- Though the case-based engine is focusing on the similarity between two cases, we have to develop a mechanism to contrast two cases [Rissland *et al.* 1987],[Rissland *et al.* 1989]. By comparing two inference trees, it is possible to construct a debate system.

- To describe legal rules in detail, we have to integrate an extended logic system such as the logic of belief and knowledge with temporal logic on MGTP.

- To improve the power of the similarity based matching of the case-based engine, we have to introduce a derivational analogy mechanism.

- As inference trees are not suitable for allowing lawyers to understand the inference steps, they are represented in natural language.

References

[Uchida *et al.* 1988] Shunichi Uchida et al. . Research and Development of the Parallel Inference System in the Intermediate Stage of the FGCS Project. In *Proc. Int. Conf. on Fifth Generation Computer Systems*, ICOT, Tokyo, 1988. pp.16-36.

[Goto *et al.* 1988] Atsuhiro Goto et al. . Overview of the Parallel Inference Machine Architecture. In *Proc. Int. Conf. on Fifth Generation Computer Systems*, ICOT, Tokyo, 1988. pp.208-229.

[Chikayama *et al.* 1988] Takashi Chikayama et al. . Overview of the Parallel Inference Machine Operating System (PIMOS). In *Proc. Int. Conf. on Fifth Generation Computer Systems*, ICOT, Tokyo, 1988. pp.230-251.

[Nitta *et al.* 1991] K. Nitta et al. . Experimental Legal Reasoning System on Parallel Inference Machine. In *Proc. PPAI Workshop of 12th IJCAI*, Sydney, Australia, 1991. pp.139-145.

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

[Rissland *et al.* 1989] E.L. Rissland et al. . Interpreting Statutory Predicates. In *Proc. Int. Conf. on Artificial Intelligence and Law*, Vancouver, CANADA, 1989. pp.46-53.

[Sartor 1991] G. Sartor. The structure of Norm Conditions and Nonmonotonic Reasoning in Law. In *Proc. Int. Conf. Artificial Intelligence and Law*, Oxford, UK, 1991. pp.155-164.

[Branting 1989] L.K.Branting. Representing and Reusing Explanations of Legal Precedents. In *Proc. Int. Conf. on Artificial Intelligence and Law*, Vancouver, CANADA, 1989. pp.103-110.

[Sanders 1991(a)] K.Sanders. Representing and reasoning about open-textured predicates. In *Proc. Int. Conf. on Artificial Intelligence and Law*, Oxford, UK, 1991. pp.137-144.

[Sanders 1991(b)] K.Sanders. Planning in an Open-Textured Domain. A Thesis Proposal. Technical Report CS-91-08, Brown University, 1991.

[Fujita *et al.* 1991] H.Fujita et al. . A Model Generation Theorem Prover in KL1 Using a Ramified-Stack Algorithm. ICOT TR-606. 1991.

[Inoue *et al.* 1991] K.Inoue et al. . Embedding negation as failure into a model generation theorem prover. ICOT TR-722. 1991.

[Allen 1984] J.F.Allen. Towards a general theory of action and time. *Artificial Intelligence*, Vol. 23, No.2 (1984),pp.123-154.