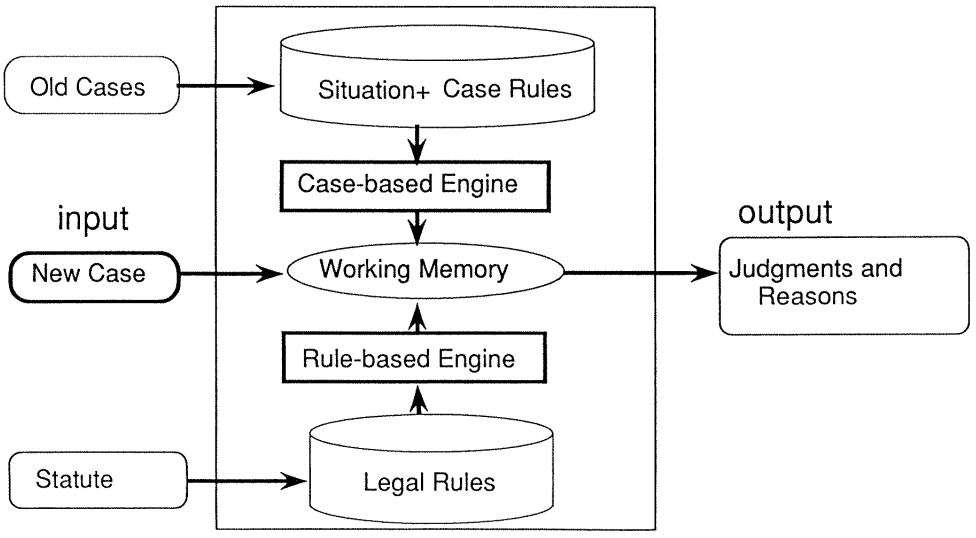
A Parallel Legal Reasoning System : HELIC-II

ABSTRACT

Legal reasoning can be modeled as a mixed paradigm of rule-based reasoning and case-based reasoning. Based on this model, we have developed a legal reasoning system, HELIC-II, on the parallel inference machine. HELIC-II draws legal conclusions by referring to statutes and old cases.

KEY FEATURES

- **A Mixed Paradigm**
  HELIC-II consists of a rule-based engine and a case-based engine. These engines draw legal conclusions cooperatively.
- **Parallel Rule Based Engine**
  The rule-based engine refers to legal rules and draws legal consequences deductively. This engine is based on the parallel theorem prover MGTP (Model Generation Theorem Prover) and has several new functions.
- **Parallel Case Based Engine**
  The case-based engine generates legal concepts by referring to similar old cases. This engine searches for similar cases and new arguments are constructed by applying case rules to the new case. Case rules are applied by matching semantic networks in parallel.



ABSTRACT

Legal knowledge consists of statutes and old cases. As a statute is a set of legal rules, inference by a statute is realized as rule-based reasoning. However, legal rules often contain legal predicates (legal concepts). Legal concepts are ambiguous and their strict meanings are not fixed until the rules are applied to actual facts. To apply legal rules to actual facts, we need rule interpretation and matching between legal concepts and concrete facts. To realize this, old cases are often referenced and their explanations are reused. Consequently, legal reasoning can be modeled as a mixed paradigm of rule-based reasoning and case-based reasoning. However, it takes this model a long time to search for similar cases and to draw conclusions, and a complex mechanism is needed to manage several inference engines. To solve these problems by parallel inference, we developed a legal reasoning system, HELIC-II, on the parallel inference machine.

Overview of HELIC-II

HELIC-II draws legal conclusions for a given case by referencing the relevant statute and old cases and outputting them in the form of inference trees. HELIC-II consists of a rule-based engine and a case-based engine. The rule-based engine refers to legal rules and draws legal consequences logically. Following is a legal rule for *manslaughter caused by negligence*. Every legal rule can be represented with this kind of inference rule.

$$
\begin{array}{l}
manslaughterCausedByNegligence(\text{“}comment\text{”}, [article = 210], \\
\quad [person(A, []), person(B, []), \{\{A \backslash = B\}\}, \\
\quad\quad action(\_action, [agent = A]), \\
\quad\quad negligence(\_negligence, [agent = A, action = \_action]), \\
\quad\quad causation(\_causation, [cause = \_action, result = \_death2]), \\
\quad\quad death(\_death2, [agent = B])] \\
\quad\quad \rightarrow \\
\quad [[manslaughterCausedByNegligence \\
\quad\quad\quad (\_, [agent = A, action = \_action])]]).
\end{array}
$$

This rule contains the legal concept of “negligence”. Whether this “negligence” has occurred or not is the key problem that must be judged for each case that is like the following case of Mary.

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. Bill found Tom crying on the street and started to drive Tom by car to the police station. However, Bill caused an accident on the way to the police station and Tom was injured. Bill thought that Tom had died of the accident and left Tom on the street. Tom froze to death.

In this case, judging negligence in mary's action is not obtained by rule-based reasoning. Such a judgement is derived from case-based inference in HELIC-II. The case-based engine generates legal concepts (e.g. negligence) from given facts by referring to similar old cases (Fig. 1).

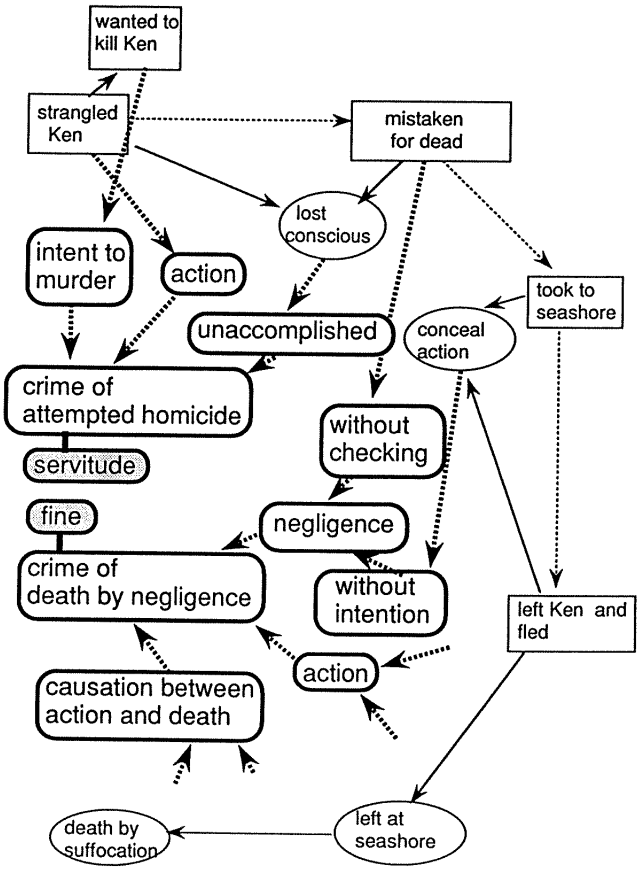


Fig.1 Arguments of Jane's Case

Fig.2 is one of outputs from Mary's case by HELIC-II.

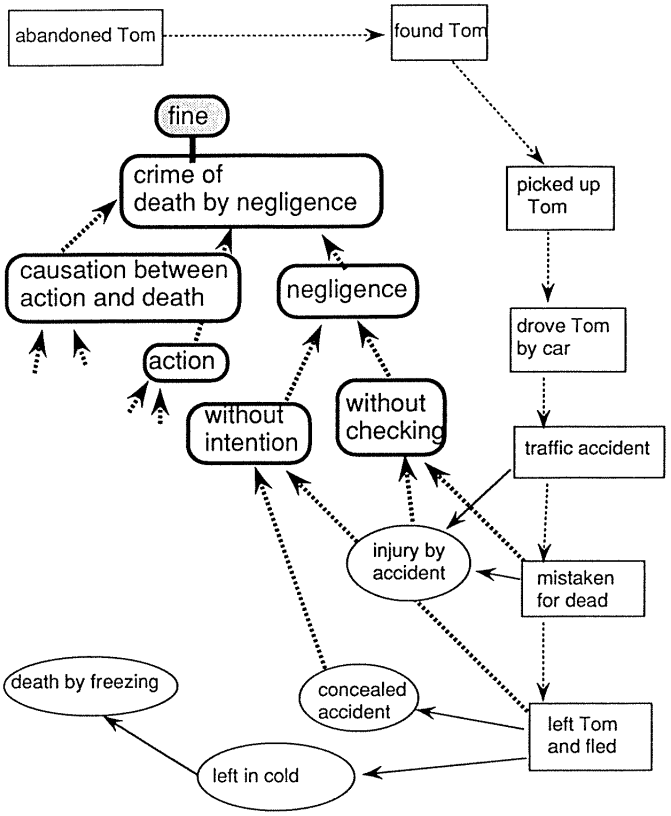


Fig.2 Arguments of Mary's Case

Conclusions are given to the user as natural language sentences, as in the following example.

> Ken's state [faint_1] and Tom's state [injury1] are similar. Jane deserted Ken, and Bill deserted Tom. Jane did not verify Ken's state [faint_1] . Similary, Bill did not verify Tom's state [injury1]. Jane deserted Ken without intending Ken's death, and Bill deserted Tom without intending Tom's death. It was insisted that Jane deserted Ken by negligence in this case. Therefore, it should be judged that Bill deserted Tom by negligence.

Rule-based inference

The rule-based engine draws legal consequences by applying forward reasoning to legal rules. As there are many legal rules, a fast rule-based engine is needed. Moreover, legal rules sometimes have exceptional rules, the rule-based engine has to include some mechanism to handle nonmonotonic reasoning. The rule-based engine is based on the parallel theorem prover MGTP (Model Generation Theorem Prover). Given a set of non-Horn clauses, MGTP generates models which satisfy all input clauses by parallel inference. To use MGTP as a rule-based engine for legal rules, and to obtain high performance by the pipeline effect, we added the following extended functions to the original MGTP.

1. **Realization of "negation as failure":** Legal rules contain two types of negations (*logical negation* and *negation as failure*). As the original MGTP could treat only *logical negation*, we enabled the new MGTP to treat *negation as failure*.

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 its initial model. The case-based engine may generate data which conflicts with each other, such as the opinions of plaintiffs and defendants. Therefore, we developed a function to split the model when such predicates reach the rule-based engine so that models don't contain conflict.

Case-based inference

A judicial precedent (old case) consists of the arguments of both sides, the opinion of the judges and a final conclusion. We represent an old case as a *situation* and some *case rules*. A *situation* contains information on the occurrences of the case and represents a set of events/objects and their *temporal relations*. Arguments by both sides are represented as a set of *case rules*.

The case-based engine generates legal concepts by referring to similar old cases. In the first stage, the engine searches for similar cases from the case base. In the second stage, new arguments are constructed by applying the case rules of selected cases to the new case. Each case rule is fired by similarity based matching.

Performance by Parallel Inference

Fig.3 shows the speedup of two inference engines. We obtained more than 50-fold speedup using the 64PEs of the Multi-PSI.

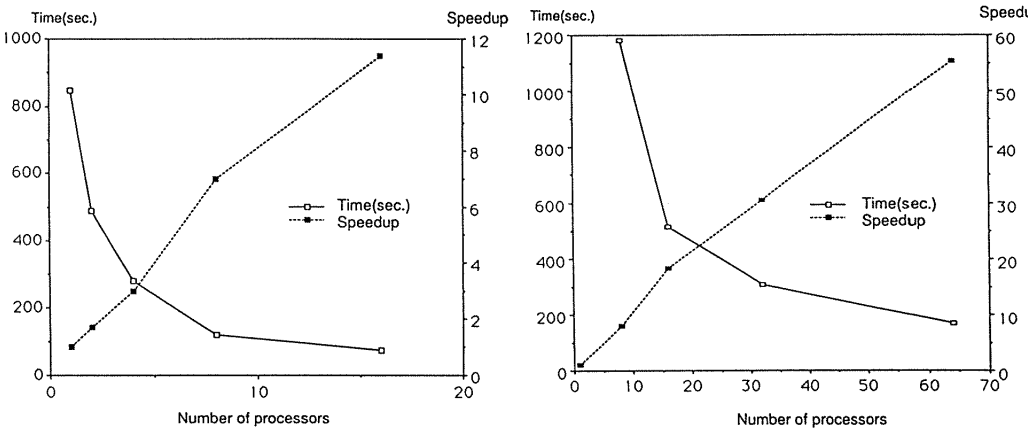


Fig.3 Performance of the rule-based engine and the case-based engine

Outline of demonstration

We demonstrate how the parallel legal reasoning system HELIC-II solves Mary's case. First, representation of Mary's case is explained. Second, the consequences drawn by HELIC-II are shown. Third, an inference tree of a consequences is explained. Fourth, more detail is explained using natural language style output.