TR-0850

A Deductive Object-Oriented Database System for Situated Inference in Law

by

S. Wong & S. Tojo (MRI)

ICOT

Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan

(03)3456-3191 ~ 5

Institute for New Generation Computer Technology

1 Introduction

Artificial Intelligence (AI) and Law is a new research field which has attracted researchers from both the legal and AI domains. Legal scholars are interested in applying AI technology to represent and study jurisprudential theories and to develop computer applications to enhance the day-to-day practices of laywers, judges, and other legal decision makers. AI researchers seek to address some of AI's fundamental issues through the formal analysis of legal concepts and precedents as well as by experimentation on knowledge system prototypes. These issues include the interpretation of open-textured concepts, reasoning by cases and rules, creating computational decision making models that embody the norms of society, and drawing arguments under opposing viewpoints and different situations. The legal domain offers a rich knowledge repository of human intellectual endeavor, which is well documented in legal articles and textbooks, to study such issues.

Typically, a legal reasoning system draws arguments by interpreting judicial precedents (old cases) or statutes (legal rules) encoded in its knowledge base, and a more advanced system includes both kinds of knowledge. Surveys and digests on the leading projects can be found in [46, 47, 51, 21, 4] (This field has been experiencing steady growth recently. For example, the biannual International Conference of AI and Law has been held since 1987, and two new journals, *Journal of Artificial Intelligence and Law* and *Law, Computers, and Artificial Intelligence*, were launched in 1992.) Today, however, research into AI and Law remains small-scale. Most implementations are written in AI programming languages, such as Prolog or Lisp, and contain only small sets of cases and rules. They do not have the capability to access and manipulate large amounts of data and lack database management services such as concurrency control, nested transactions, and data persistence. Reasoning in law, however, is a knowledge-intensive endeavor. The lack of appropriate tools to scale up the legal knowledge bases of these prototypes is a major handicap to the growth and potential contributions of this interdisciplinary field. On the other hand, the database (DB) community has yet to develop tools which are expressive enough to satisfy the knowledge modeling needs of the AI and Law researchers.

Taking the AI point of view, legal reasoning systems have been a key research activity in the Fifth Generation Computer System (FGCS) project [42, 43, 59]. The legal domain of concern is penal code. This project has devised a formal model of legal argumentation, $\mathcal{SM}$, [58] and has developed a Deductive Object-Oriented Database (DOOD) System, $\mathcal{QUIXOTE}$ [61], whose representation language can map the formulation into a computational form on the Parallel Inference Machines (PIM) [53]. The legal knowledge representation is based on situation theory [8, 7], a branch of the philosophy of language that is adapted to formalize legal rules and precedents and to reason about vague concepts and arguments. Our legal reasoning prototype, based on this formulation, includes a control program and a set of knowledge bases. The control program is written in the parallel logic programming language, KL1 [12]. The set of knowledge bases includes a dictionary of legal ontologies, a database of old cases, and a database of statutes.

In this paper, we focus on the formal theory and data modeling aspects of our legal reasoning research. We discuss the specific features of the $\mathcal{QUIXOTE}$ system, that can be used to support situated inference and to manage legal databases of various sorts. In addressing the complex issues of AI and Law, this study has brought together two previously unrelated fields, deductive object-oriented database and situation theory. We believe that the principles and techniques discussed can be extended to many other demanding applications. On the other hand, this work, to our knowledge, is the first attempt to provide an advanced knowledge base management system tool, which includes database management functions and a single language for both database and programming purposes, to build large knowledge systems for advanced legal reasoning applications.

The organization of this paper is as follows. Section 2 describes the modeling of legal knowledge and reasoning at the abstraction level, using the theory of situations. Section 3 discusses the realization of this formulation at the database level using $\mathcal{QUIXOTE}$. Section 4 illustrates situated inference mechanisms supported by this database system, and presents legal examples. We discuss the related work in Section 5 and conclude this paper in the last section.

2 Formal Representation of Legal Knowledge

This section introduces a formal model for legal reasoning, especially, penal code, at an abstraction level. As the formulation is based on *situation theory*, we call it a *situation-theoretic model* ($\mathcal{SM}$). This model offers definitions and conditions that not only help to carry out a study of legal concepts within the framework of mathematics but also guides the design of legal reasoning systems.

We adopt situation theory, instead of classical logic, because its semantics better captures the notion of *open-texture*. A legal concept exhibits open texture in that it is precisely defined only for those cases which have been decided by a court, and there is no precise definition of the concept for cases which have still to come to court. The interpretation of such vague and discretionary concepts in law depends on the situations surrounding new cases. Many problems in natural language understanding are also ascribed to such situation dependency, and various semantics have been proposed, for example, as successors to possible world semantics [18] (the first formalization of situatedness), such notions as situations [7], mental spaces [19], and DRT [27] have been introduced. Although all of these semantics have different objectives and philosophies, they share many similarities. One of the advantages of situation theory is its uniform way of representing various kinds of *situatedness*, that is, $s \models \sigma$, the interpretation of a phrase or sentence, σ, under the scope of a situation s.

Various proponents of situation theory do not provide clear definitions on what constitutes a situation. Consequently, the term situated inference has a number of meanings, with most of them impractical for designing computing systems. The observation of this study is that situations in law can be defined abstractly in terms of a set of infons or sentences about a case. $\mathcal{SM}$ deals with such abstract situations. The presumption is that abstract situations and the constraints between them would be adequate to describe the *logical* flow of information in real situations [16] and would therefore be userful to the design of legal reasoning systems.

2.1 General Terms

The ontologies of $\mathcal{SM}$ include objects, parameters, relations, infons, and situations. An object designates an individuated part of the real world: a constant or an individual in the sense of classical logic. A parameter refers to an arbitrary object of a given type. An n-placed relation is a property of an n-tuple of argument roles, $r_1, \cdots, r_n$, or slots into which appropriate objects of a certain type can be anchored or substituted. For example, we can define 'eats' as the four-place relation of action type as:

$$< \text{eats:}\mathit{Action} \mid \mathit{eater{:}ANIMAL,\ thing\text{-}eaten{:}EDIBLE\text{-}THING,\ location{:}LOC,\ time{:}TIM} >$$

where *eater, thing-eaten, location,* and *time* are roles and the associated types, *ANIMAL* denotes the type of all animals, *EDIBLE-THING* denotes the type of all edible substances, *LOC* and *TIM* are types of spatial and temporal location.

An infon σ is written as $\ll Rel, a_1, ..., a_n, i \gg$, where Rel is a relation, each argument term a_i is a constant object or a parameter, and i is a polarity indicating 1 or 0 (true or false). (This notation is intended to emphasize that 'infons' are semantic concepts, not a syntactic representation.) If an infon contains an n-place relation and m argument terms such that m < n, we say that the infon is *unsaturated*. IF m = n, it is *saturated*. Any object assigned to fill an argument role of the relation of that infon must be of the appropriate type or must be a parameter that can only anchor to objects of that type.

Further, an infon that has no free parameters is called a *parameter-free* infon; otherwise, it is a *parametric* infon. If σ is an infon and f is an *anchor* for some or all of the parameters that occur free in σ, we denote, by $\sigma[f]$, the infon that results by replacing each v in the domain of f that occurs free in σ with its value (object constant) $f(v)$. If I is a set of parametric infons and f is an anchor for some or all of the parameters that occur free in I, then $I[f] = \{\sigma[f] \mid \sigma \in I\}$. In addition, an abstract situation is said to be *coherent* if it does not support both an infon and its negation. Two abstract situations s and s' are said to be *compatible* if their union is a coherent situation. The situations within a legal case are presumed to be compatible with one another, but no such presumption can be made across different cases.

A $\mathcal{SM}$ is a triplet $\langle \mathcal{P}, \mathcal{A}, \models \rangle$, where $\mathcal{P}$ is a collection of abstract situations including judicial precedents, a new case, c_n, and a world, w, that is a unique maximal situation of which every other situation is a part; $\mathcal{A}$ are the defendant and plaintiff agents; and $\models$ is the support relation. The latter satisfies the following conditions [16]:

Condition 2.1 (Supports Relation)
i. For any $s \in \mathcal{P}$, and any atomic infon σ, $s \models \sigma$ if and only if (iff) $\sigma \in s$.

ii. For any s, any σ, β,

- For any s that contains (as constituents) all members of u, $s \models (\exists \dot{x} \in u)\sigma$ iff there is an anchor, f, of a parameter, $\dot{x}$, to an element of u, such that $s \models \sigma[f]$.
- For any s that contains all members of u, $s \models (\forall \dot{x} \in u)\sigma$ iff for all anchors, f, of $\dot{x}$ to an element of u, we have $s \models \sigma[f]$.

iii. For any $s \in \mathcal{P}$, and any set of infons I, $s \vdash I$ if $s \vdash \sigma$ for every infon σ in I. $\square$

Thus, the notation $s \models \sigma$ denotes a proposition about σ whose truth values are situation-dependent and may be at issue, whereas $w \models \beta$ asserts that β is universally true. In addition, let v be a parameter. By a *condition* on v we mean any finite set of parametric infons. (At least one of these should involve v, otherwise, the definition is degenerate). We define a new parameter, $v \| C$, called a *restricted parameter*. The idea is that $v \| C$ will denote an object of the same basic type as v, that satisfies the requirements imposed by C. This amounts to our placing a more stringent requirement on anchors. Such a notion of restricted parameter enhances the expressive power of knowledge representation.

2.2 Concept Matching

We introduce certain specific terms, *relevance level, infon matching*, and *situation matching*, to extend the general $\mathcal{SM}$ terms into the legal domain. In a legal event, an agent would consider some facts (infons) to be more relevant than others in reaching an argument. To estimate such weighting on facts, $\mathcal{SM}$ assigns every infon in an old case with a level of relevance. For example, the restricted parameter $\dot{\sigma} = \sigma \| \ll \text{relevance-level}, \sigma, \lambda, 1 \gg$, where λ denotes a certain weight of relevance.

One distinction of legal reasoning is the matching of the new facts with those of precedents to generate similar arguments which may hold in the new case [42, 4]. No two events are exactly alike, but the idea of precedent-based matching presupposes that a prior decision will control subsequent facts that are like the first. Yet, given the lack of absolute identity, the decision-maker of the new case must evaluate the determinant of likeness. To this end, $\mathcal{SM}$ adopts a concept of structural matching. Since cases are composed of infons, the model first defines the matching relation between these basic units of information. Note that a case infon is always parameter-free.

Condition 2.2 (Infon Matching)
For c_n and an old case c_o, $\sigma_n = \ll Rel_1, a_1, ..., a_n, i_1 \gg \in c_n$, $\sigma_o = \ll Rel_2, b_1, ..., b_m, i_2 \gg \in c_o$,
(a) (Exact): $\sigma_n \simeq_{iem} \sigma_o$ iff (i) m = n; (ii) $i_1 = i_2$; (iii) Rel_1 and Rel_2 are of the same type; (iv) for every argument a_i of a non-infon type, there exists b_k which is of the same role or type and has not been matched with another argument; (v) for every a_j of an infon type, there exists b_y that satisfies the same set of conditions.

(b) (Partial): $\sigma_n \simeq_{ipm} \sigma_o$ if m $\leq$ n and all argument terms of σ_o are matched. $\Box$

where a Rel b intends to denote $w \models \ll Rel, a, b, 1 \gg$. Condition 2.2.a is strict and difficult to fulfull. Condition 2.2.b relaxes the matching of roles to be partial. Note, such a *relaxation* is predetermined by and varied with old cases. Clearly, $\simeq_{iem}$ is an equivalence relation while $\simeq_{ipm}$ is an asymmetric relation. Infon-matching relations are the building blocks for defining situation-matching relations.

Condition 2.3.a (Exact Situation Matching)
For any $s_n \subseteq c_n$, $s_o \subseteq c_o$, $s_n \simeq_{sem} s_o$ iff for every σ of $s_o \vdash \sigma$, there exists ρ of $s_n \models \rho$ such that $\sigma \simeq_{iem} \rho$, and vice versa. $\Box$

Given the ambiguity and complexity of legal phenomena, it is rare to find an exact match between cases as depicted. One heuristic is to search for an old case whose sets of important propositions match those of the pending case. This is called the *partial matching* of situations. Important propositions of an old case are identified by their relevant levels in that case.

Condition 2.3.b (Partial Situation Matching)
For any $s_n \subseteq c_n$, $s_o \subseteq c_o$, $s_n \simeq_{spm} s_o$ iff for every σ of $s_o \models \sigma || \ll$relevance-level, σ, l,1 $\gg$ s.t. $l \geq l_i$, there exists ρ of $s_n \models \rho$ s.t. $\sigma \simeq_{ipm} \rho$. $\Box$

When there is no danger of confusion, we shall write $\simeq_s$ to denote a matching relation between situations and $\simeq_i$ between infons. Obviously, $s_n \simeq_{sem} s_o$ implies $s_n \simeq_{spm} s_o$.

2.3 Situated Inference Rules

Legal reasoning is a rule-based decision-making endeavor. But, the structure of a rule and it does are complex. Opposing agents extract different information from a common set of cases and statutes to support their arguments. Thus, the debate between the two agents is *goal-oriented*. Nevertheless, the information that an agent can pick up and the credibility of such information depends on the situations and hypotheses. Figure 1 illustrates an AND-OR inference tree with a goal as the top node and a set of initial facts and hypotheses as leaf nodes. An arc indicates the AND path. The link can be bidirectional, depending on whether forward reasoning or backward reasoning is used to derive the goal.

Typically, a legal inference tree can be partitioned into four layers. The bottom layer consists of a set of basic facts and hypotheses, the second involves case rules of individual precedents, the third involves case rules which are induced from several precedents or which are generated from certain legal theory, and the top layer concerns legal rules derived from statutes. An individual or local case rule is used by an agent in an old case to derive plausible legal concepts and propositions. These rules vary from case to case, and their interpretation depends on particular views and priorities. An induced case

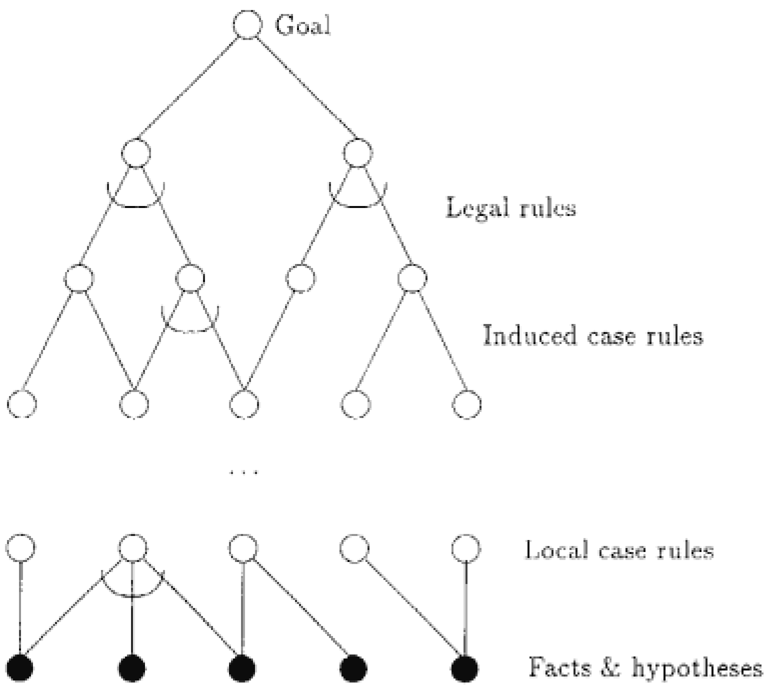

Figure 1: A legal inference tree

rule has a broader scope and is generalized from a set of precedents. Legal rules are general provisions and definitions of crimes. They are supposed to be universally valid (in the country where they are imposed) and neutral. That is, the applicability of these rules is independent of the view of either side (plaintiff or defendant) and every item of information (infon) included is of equal relevance. Further, though it rarely happens, it may be possible for an agent to skip one or two case rule layers in attaining a legal goal.

Let us denote $c \models I$ as a shorthand for $c \models \sigma_1, c \models \sigma_2, ..., c \models \sigma_n$, where $\sigma_0, \sigma_1, ..., \sigma_n$ are infons, then in the rule-oriented legal domain, the general form of *situated inference* is as follows.

Rule 2.1 (General Rule)
Let $I_0, I_1, ..., I_n$ be sets of infons, and $s_0, s_1, ..., s_n$ be situations, $s_0 \models I_0 \Leftarrow s_1 \models I_1, s_2 \models I_2, ..., s_n \models I_n/B$. $\Box$

This rule can be read as: "if s_1 supports I_1, s_2 supports I_2, and so on, then we can infer that s_0 supports I_0 under the background conditions or constraints B." $s_0 \models I_0$ is called the head of the rule and the other called the body of the rule. The background conditions, B, are required to be coherent

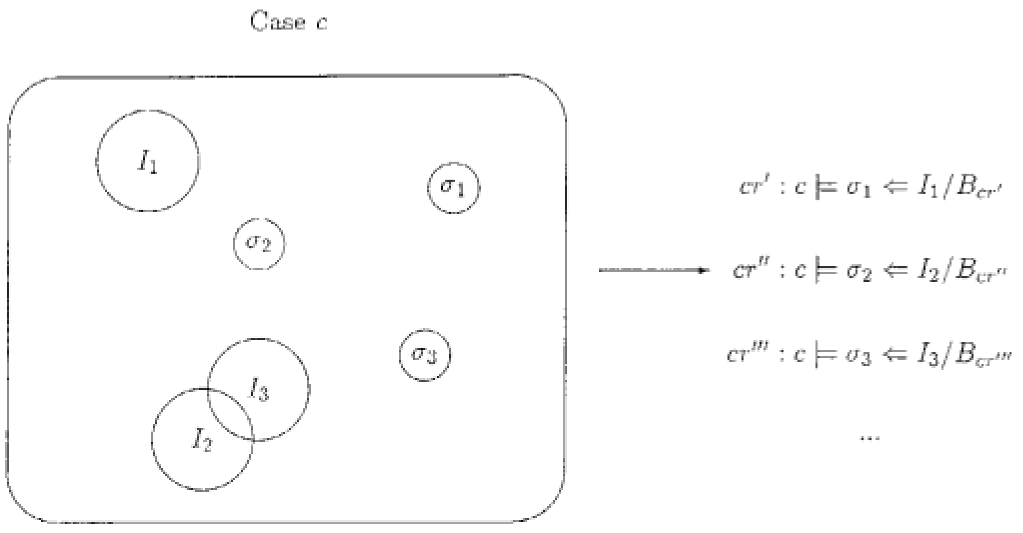
Case c
I_1
σ_1
σ_2
σ_3
I_3
I_2
$cr' : c \models \sigma_1 \Leftarrow I_1/B_{cr'}$
$cr'' : c \models \sigma_2 \Leftarrow I_2/B_{cr''}$
$cr''' : c \models \sigma_3 \Leftarrow I_3/B_{cr'''}$
...

Figure 2: A case and its set of local rules.

and satisfied before execution of the rule. We are particularly interested in three rule instances: local case rules, induced case rules, and legal rules. A local case rule is as follows:

Rule 2.2 (Local Rule)
For $c \in \mathcal{P}$, $cr: c \models \sigma \Leftarrow c \models I/B_{cr}$. ☐

where I is called the antecedent of the rule, σ is the consequent infon, and cr is the label of the rule, which is not itself part of the rule but which serves to identify the rule. Sometimes, we simply write cr: $c \models \sigma \Leftarrow I/B_{cr}$. Both σ and I are parameter-free. One unique feature of inference rules in law is that the consequent is not disjunctive and often a single predicate. The reliability and the scope of application of a local rule will be subject to a set of *background conditions*, B_{cr}. The conditions include information such as an agent's goal and hypotheses; these are crucial in debate to establish the degree of certainty and the scope of applicability of that rule. Usually, it becomes necessary to take background conditions into account and investigate what they are when the conclusion drawn from the case rule leads to conflict with others or a change in circumstances that weakens the applicability of that rule. Many case rules exist in one case and often yield incompatible conclusions. But, the background conditions make clear their hypotheses and perspective. When there is no danger of conclusion, we will write such a rule without stating its background conditions.

The local rules of an old case can be considered as being logical extensions of that case (remember that a case situation is composed of a set of infons in $\mathcal{SM}$). In Figure 2, the antecedents of rules cr'' and cr''' are overlapping, but different conclusions are drawn. Some of these rules were accepted by the judge while some were not. However, the failed rules of an old case might provide a useful insight in the new case, as the situation and the judge are now different. In addition, a series of local rules for the same case may be 'hooked' together by the notion of *serial composition* [16], That is, given $cr_1 : c \models \sigma_0 \Leftarrow \sigma_1/B_1$, ..., $cr_n : c \models \sigma_{n-1} \Leftarrow \sigma_n/B_n$, we have $cr : c \models \sigma_0/B$, where $cr = cr_1; cr_2; ...; cr_n$ to indicate the ordering of serial composition. The set of background conditions of the composite rule is a union of the background conditions of the component rules, i.e., $B = B_1 \cup B_2 ... \cup B_n$ and is coherent; otherwise the composition would not be permissible. As an example, a court case is usually set up with two conflicting perspectives: plaintiff and defendant, and the background conditions of any case rule can embrace only one of them. Composing rules having different perspectives is thus not permissible.

Another form of case rule is generalized or induced from several precedents. Owing to its generic nature, an induced case rule is represented as a constraint between two parametric situations, rather than parameter-free situation types. Denote I' and σ' as two sets of parametric infons such that all parameters that occur in the latter also appear in the former. An induced rule is written as:

Rule 2.3 (Induced Rule)
For any $c_1, ..., c_k \in \mathcal{P}$, $c = c_1 \cup c_2, \cup ... \cup c_k$, $ir : c \models \sigma' \Leftarrow I'/B_{ir}$. $\Box$

where c is coherent and ir is the rule label. Similarly, a legal rule is:

Rule 2.4 (Legal Rule)
$lr : w \models \sigma' \Leftarrow I'/B_{lr}$. $\Box$

where lr is the rule label and B_{lr} states the background legal theory, such as an aim of punishment or that of crime prevention, but not both. Such information is crucial in interpreting the antecedent infons.

2.4 Substitution

A critical process of legal reasoning is to decide which legal rules apply to a new case. For example, the prosecutor is required to establish *criminal intent* and *causality* between the defendant's action and its consequence in order to apply legal rules. On the other hand, the defense lawyer tries to refute all such claims of the prosecutor. Since there are no fixed rules to define the intent and causality, lawyers on both sides normally seek hints from old cases. This process is modeled as the interaction between the matching and substitution conditions.

When a situation of a new case, c_n, supports a similar antecedent of a local rule of c_o, one can draw a conclusion about the new case similar to the consequent of that rule. That is,

$$\begin{tikzcd}
I' \arrow[r, "\overset{cr^s}{\Longrightarrow}"] \arrow[d, leftrightarrow, "\simeq_s"'] & \theta\sigma \\
I \arrow[r, "\overset{cr}{\Longrightarrow}"] & \sigma
\end{tikzcd}$$

Figure 3: Case substitution.

Rule 2.5 (Local Rule Substitution)
For $c_n, c_o \in \mathcal{P}$, $cr^s : c_n \models \sigma\theta$ if $cr : c_o \models \sigma \Leftarrow I/B_{cr}$ and $c_n \models I'/\{B_{cr}\theta \cup B_n\}$ such that $I' \simeq_s I$. □

where cr^s is the label of the new rule, B_n is the original background of the new case, I', and the combined condition after the substitution, $B = B_{cr}\theta \cup B_n$, is coherent. The function θ forms a *link* that connects c_n with c_o. This function replaces all terms (objects and relations) in σ and B_{cr} that also occur in I with their matched counterparts in I'. Figure 3 presents a diagram of such a substitution, that does not include the background conditions. Note that $c_n \models I'/B$ implies that $c_n \cup B \models I'$. In addition, referring to Rule 2.5, the substitution merely replaces terms and does not change the polarities of infons. Also, the information of case matching B_n is not related to $B\theta$ and thus does not create compatibility problems. Thus, it follows that $\{c_n \cup B\}$ is coherent.

In a court case, both sides (plaintiff and defendant) are normally ignorant about the assumptions and hypotheses of each other's claims. An essential technique, used to reveal such 'hidden' information, is cross-examination. Incorporating the background conditions into legal constraints (case and legal rules) allows us to capture this essential feature of legal reasoning for knowledge-based applications. Given any $c \in \mathcal{P}$ such that $c \models I_1/B_1$ and $c \models I_2/B_2$, if $I_1 \cup I_2$ is incoherent, then B_1 and B_2 become incompatible with each other. The reason is that $c \cup B_1 \models I_1$ and $c \cup B_2 \models I_2$ imply $\{c \cup B_1\} \cup \{c \cup B_2\} \models I_1 \cup I_2$, and if $I_1 \cup I_2$ contains some σ and its negation, $\overline{\sigma}$, this means that $\{c \cup B_1\}$ is incompatible with $\{c \cup B_2\}$. Subsequently, B_1 and B_2 are incompatible.

Thus, to combine the conclusions supported by different situations, the background conditions of both conclusions is required to be compatible. For this reason, the background conditions of Rule 2.1 must be coherent. Rather than substitution, a consequent is derived from a legal rule (see Figure 4) or

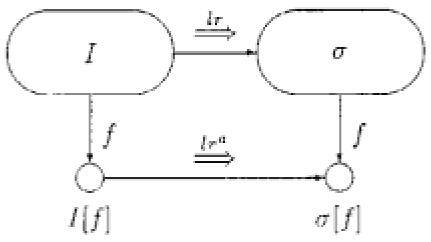

Figure 4: Legal anchoring.

an induced rule via anchoring.

Rule 2.6 (Induced Rule Anchoring)
For $c_n, c_1, ..., c_k \in \mathcal{P}$, such that $c = \{c_1, c_2, ..., c_k\}$, $ir^a : c_n \models \sigma[f]$ if $ir : c \models \sigma \Leftarrow I/B_{ir}$ and $c_n \models I[f]/\{B_{ir}[f] \cup B_n\}$ $\square$

Rule 2.7 (Legal Rule Anchoring)
For $c_n \in \mathcal{P}$, $lr^a : c_n \models \sigma[f]$ if $lr : w \models \sigma \Leftarrow I/B_{lr}$ and $c_n \models I[f]/\{B_{lr}[f] \cup B_n\}$ $\square$

In this section, we introduced the $\mathcal{SM}$ definition for situation matching and legal implications. Armed with this technique, we can represent, diagramatically, the flow of information in a goal inference tree of Figure 1. In Figure 5, we show such a legal inference in the forward reasoning manner. For simplicity, this inference involves only local and legal rules. The black circles, I_1', I_2', and σ, denote the situations of a new case, c_n while situations I_1 and I_2 are of old cases. Two immediate arguments, β_1 and β_2 are drawn using local rules cr_1 and cr_2. Together with σ, the goal $\gamma[f]$ is anchored or attained by applying the legal rule lr. From the coherency condition of Proposition 2.4, it follows that all concepts of a single goal tree must share the same legal perspective: the plaintiff's or the defendant's. This figure indicates that the matching relation of I_1 is stronger than that of I_2. One can also probe into background conditions, linked by appropriate rule labels, of these arguments to retrieve the underlying hypotheses and legal theories.

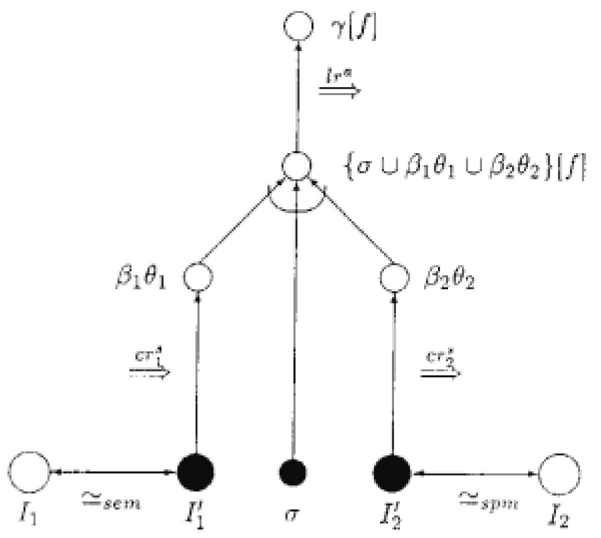

Figure 5: A visual representation of legal inference.

3 Data Modeling of Legal Knowledge in *QUIXOTE*

This section introduces the *QUIXOTE* language and shows how this language can be used to map the *SM* concepts in computable form for legal reasoning applications. This *QUIXOTE* language [61, 62] is a hybrid of the deductive object-oriented database (DOOD) language and constraint logic programming (CLP) language. It resembles F-logic [28], in that it is a DOOD language that includes powerful extensions into logic programming such as subsumption, complex objects, and modules. Compared with conventional CLP language, *QUIXOTE* has a symbolic constraint domain which makes it suitable for describing the legal situations depicted in legal documents written in, albeit tight and formal, natural language.

A typical *QUIXOTE* database includes the following: (i) the subsumption relations among basic objects, (ii) the submodule relations among modules, and (iii) rules. As stated in Section 1, our legal reasoning system consists of three databases of distinct knowledge contents, namely, a dictionary, a case base, and a statute base. Accordingly, we first introduce the *objects* and *modules* of *QUIXOTE*, explain the data structure of a legal dictionary, and then describe the use of *QUIXOTE rules* to represent case-based rules and statutes.

In *QUIXOTE*, the concepts of situation theory are rephrased as follows:

situation theory	$\mathcal{QUIXOTE}$
situation	module
infon	attribute term
relation name	basic object
type	subsumption
role	label
supporting relation ($\models$)	membership in module ($:$)

3.1 Objects, Modules, and Matching

3.1.1 Object terms

The legal dictionary has two parts: the *concept lattice* and the definition of *relations* (viz. the tuples of roled slots with predicates). We first introduce the $\mathcal{QUIXOTE}$ notion of *objects* and *subsumption* relation for forming the concept lattice, and, thereafter, we describe $\mathcal{QUIXOTE}$'s *attribute terms* to represent the relations and infons of situation theory.

Object terms, *Obj*, in $\mathcal{QUIXOTE}$ consist of a set of *basic objects*, *Bobj*, a set of *complex objects*, *Cobj*, and a set of *variables*, *Var*. We denote the *subsumption* relation, $\sqsupseteq$, as a partial relation between basic objects such that for any $\mathtt{a}, \mathtt{b} \in Bobj$, $\mathtt{a} \sqsubseteq \mathtt{b}$ means that a is more specific than b, or, intuitively, a *is_a* b. The following is an example of the subsumption relations among basic objects (In $\mathcal{QUIXOTE}$ syntax, $\sqsubseteq$ is represented by '=<'.)

$$\mathtt{infant} \sqsubseteq \mathtt{person},\quad \mathtt{baby} \sqsubseteq \mathtt{person},\quad \mathtt{person} \sqsubseteq \mathtt{creature},\quad \mathtt{lion} \sqsubseteq \mathtt{creature},$$

Together with the basic objects $\bot$ (bottom) and $\top$ (top), we have

$$\forall x \in Bobj, \bot \sqsubseteq x,\ x \sqsubseteq \top.$$

Thus, a concept lattice of basic objects, $< Bobj, \sqsupseteq >$, is a finite bounded complete partial order.

A *complex object* is of the form $o[l_1 = v_1, l_2 = v_2, \cdots]$, where $o \in Bobj$ and for any $l_i \in Bobj, v_i \in Obj$. l_i is also called a *label*. The order of labels is not strict. For example, `o[l=a,m=b]` and `o[m=b,l=a]` are treated as being identical objects. The subsumption relation between basic objects is extended to the relation between complex objects, or between complex and basic objects, in the following manner.

$$\mathtt{h}[\mathtt{l}_1 = \mathtt{v}_1, \ldots] \sqsubseteq \mathtt{h}'[\mathtt{l}'_1 = \mathtt{v}'_1, \ldots] \quad \text{iff} \quad \mathtt{h} \sqsubseteq \mathtt{h}', \forall i \exists j, \mathtt{l}_j = \mathtt{l}'_i, \mathtt{v}_j \sqsubseteq \mathtt{vi}'$$

$$\mathtt{h}[\mathtt{l}_1 = \mathtt{v}_1, \ldots] \sqsubseteq \mathtt{o} \quad \text{iff} \quad \mathtt{h} \sqsubseteq \mathtt{o}$$

Similarly, the database operations, *meet* and *join*, between complex objects are defined as the greatest lower bound and least upper bound, since the basic objects compose a complete lattice. For example, the following relation holds when we have $\mathtt{strangle} \sqsubseteq \mathtt{homicide}$ and $\mathtt{poison} \sqsubseteq \mathtt{homicide}$.

$\texttt{strangle[agent = tom]} \sqsubseteq \texttt{homicide}$,

$\texttt{poison[agent = tom, coagent = mary]} \sqsubseteq \texttt{homicide[coagent = mary]}$

while there is no subsumption relation between poison[agent = tom] and homicide[coagent = mary].

An *attribute term* is an object term with attached property specifications, i.e., a set of '1=v'. Such a term for a complex object has the following form.

$$\underbrace{\overbrace{o}^{\text{basic obj.}}\ \overbrace{[l_1 = v_1, \ldots]}^{\text{intrinsic}}}_{\text{complex obj.}} / \overbrace{[l_2 = v_2, \ldots]}^{\text{extrinsic}}$$

Note that we distinguish the properties of a complex object from those of an attribute term. The former is called an *intrinsic* attribute while the latter is called an *extrinsic* attribute. The label-valued relations of attribute terms are the following three operations:

$$\begin{aligned} &\texttt{o/[l = X]} \quad \text{iff} \quad \texttt{o} \mid \{\texttt{o.l == X}\} \\ &\texttt{o/[m} \rightarrow \texttt{U]} \quad \text{iff} \quad \texttt{o} \mid \{\texttt{o.m} \sqsubseteq \texttt{U}\} \\ &\texttt{o/[n} \leftarrow \texttt{V]} \quad \text{iff} \quad \texttt{o} \mid \{\texttt{o.n} \sqsupseteq \texttt{V}\} \end{aligned}$$

where $O \mid C$ denotes an object term O with constraint C. We introduce the *dotted term* notation, O.L, where O is an object term and L is a label, to specify the value of the L (extrinsic) attribute of O. By default, the properties of an object are inherited from related objects via the subsumption relation.

$$\text{If } \texttt{o} \sqsubseteq \texttt{p}, \text{ then } \forall \texttt{l}, \texttt{o.l} \sqsubseteq \texttt{p.l}$$

For complex objects, the values of intrinsic attributes override those of extrinsic ones. As an example, when death/[cause=suicide] holds, **death**[cause=murder].cause (= murder) is not subsumed by death.cause (= suicide) although we have **death[cause = murder]** $\sqsubseteq$ **death**.

These attribute terms can be used to represent $\mathcal{SM}$ infons. Let us consider the following relation (see Section 2.1):

```
abandon/[agent=Agent, object=Coagent, place=Loc]
      | {abandon ⊑ act, Agent ⊑ human, Coagent ⊑ human, Loc ⊑ location} .
```

This is a $\mathcal{QUIXOTE}$ representation of the sentence, "Agent's act of abandoning Coagent, where both Agent and Coagent are human." The subsumption relation stated in the constraints denotes the type specification in situation theory, such that the corresponding $\mathcal{SM}$ representation is:

$$< abandon : action, \quad \dot{x}_{agt} : human, \quad \dot{y}_{obj} : human >$$

where *abandon* is of the *action* type, *agt* (agent) and *cgt* (coagent) are a type of *human*. The dictionary maintains a list of legal relations with distinct names, and the object lattice includes the subsumption hierarchy between the names of these relations. Given the uniqueness of these relation names, this lattice also implies a hierarchy between these relations.

In the following, we list some examples of infons in $\mathcal{QUIXOTE}$ format.

```
abandon/[agent=Agent, coagent=Coagent]
   | {Agent ⊑ human, Coagent ⊑ human ∪ object},
make/[agent=Agent,object=accident/[agent=Agent,of=traffic]]
   | {Agent ⊑ human},
injure/[agent=Agent, coagent=Coagent] | {Agent ⊑ human, Coagent ⊑ human},
leave/[agent=Agent, coagent=Coagent] | {Agent ⊑ human, Coagent ⊑ human},
death/[agent=Agent] | {Agent ⊑ human},
                              ⋮
```

It is worth noting that the infon in line three, `make/[agent= Agent, object= accident[agent= Agent, of= traffic]]`, is a nested infon. $\mathcal{QUIXOTE}$ supports such a recursive data structure as the value of an `object`, as the following two declarations:

```
make/[agent=Agent,object=accident] | {Agent ⊑ human},
accident/[agent=Agent,of=traffic] | {Agent ⊑ human}.
```

3.1.2 Modules

A $\mathcal{QUIXOTE}$ legal database consists of a hierarchy of modules. Each module is identified by an object term called a *module identifier* and consists of a set of rules. Like many object-oriented programming languages, the rules of one module are inherited by its submodules. The submodule relation, $\sqsupseteq_S$, is a partial relation between module identifiers that specifies rule inheritance among modules. For example, if case1 $\sqsupseteq_S$ case2, all rules and facts in module `case2` are inherited by module `case1`. (In $\mathcal{QUIXOTE}$ syntax, $\sqsupseteq_S$ is represented by '`>-`'.) We called `case1` a *submodule* of `case2` and `case2` a *supermodule* of `case1`. Module and rule inheritance are powerful devices for classifying and modeling situation-dependent knowledge.

Identical objects must have equal properties within a module, but are allowed to have distinct properties between different unrelated modules. For instance, the following piece of code is inconsistent, since the same `homicide` object has two different `agent` properties.

```
sit_1 :: homicide/[agent=tom];;
sit_1 :: homicide/[agent=mary];;
```

On the other hand, the following code is consistent, provided that `sit_1` and `sit_2` are not related.

```
sit_1 :: homicide/[agent=tom];;
sit_2 :: homicide/[agent=mary];;
```

3.1.3 Realization of Concept Matching

The concept of infon matching, stated in Condition 2.2 of Section 2.2, is realized in *QUIXOTE* as follows.

Operation 3.1 (Infon Matching)
For any two attribute terms o1 and o2,

1. If o3 exists, such that o1 $\sqsubseteq$ o3, and o2 $\sqsubseteq$ o3 in a given concept lattice, then o1 and o2 are interpreted as being *partial matched infons.*

2. If the basic object parts of two attribute terms are found to be identical, the two attribute terms are interpreted as being *exactly matched infons.* □

Under Operation 3.1, for example, abandon and leave are *partially matched* if the legal dictionary contains:

```
abandon ⊑ act, leave ⊑ act
```

and abandon/[agent=jim] is *exactly matched* with abandon/[agent=tom]. We next define situation matching in terms of modules and objects in *QUIXOTE* (see Cond. 2.3).

Operation 3.2 (Situation Matching)
For any m_1 and m_2,

1. If, for every attribute term in m_1, there is one and only one attribute term in m_2 that can match it exactly, and vice versa, then the two modules are interpreted as being *exactly matched situations.*

2. If, for any o_1 in m_1 whose *relevance* value subsumes a given object (viz. the threshold level), there is an attribute term o_2 in m_2, that can be partially matched with o_1, the two modules are interpreted as being *partially matched situations.* □

For example, if two modules contain:

```
m_n :: {abandon/[agent=mary, object=june],
        leave/[agent=mary, object=june]};;
m_o :: {abandon/[agent=jim, object=tom],
        leave/[agent=jim, object=tom]};;
```

QUIXOTE would assert that m_n is *exactly matched* with m_o. Consider another pair of descriptions:

```
m_n :: {abandon/[agent=mary], leave/[agent=mary, object=june]};;
m_o :: {abandon/[agent=jim, object=tom] | {abandon.relevance == 13},
        leave/[agent=jim, object=tom] | {leave.relevance == 12},
        poor/[agent=jim] | {poor.relevance == 11};;
```

where 11 =< 12 =< 13, we have:

- if the threshold value is 12, then m_n is *partially matched* with m_o.
- if the threshold value is 11, then m_n is *not partially matched* with m_o.

3.2 Situated Inference Rules

A *QUIXOTE* rule takes the following form (compare with Rule 2.1).

$$\overbrace{m_0 :: H}^{head} \mid \overbrace{HC}^{head\_constraints} \Leftarrow \overbrace{m_1 : B_1, \ldots, m_n : B_n}^{body} \mid\mid \overbrace{BC}^{body\_constraints} \;;;$$

where H or B_i is a literal while HC and BC are sets of subsumption constraints. An object term, m_i, is called a *module identifier*. The above rule exists in the module m_0. Intuitively, this means that if every B_i holds in a module m_i under the constraints BC, then H and constraints HC hold in m_0, where H and B_i's are *object terms* or *attribute terms*. HC works as constraints in the sense of conventional CLP language[24], while BC is processed abductively. Constraints in *QUIXOTE* are sets of formulas in terms of a subsumption relation among object terms and dotted terms. Each formula has the form $<term>, <op>, <term>$ where $<term>$ is an object term or a dotted term and $<op>$ is $=$, $\sqsubseteq$, or $\sqsupseteq$. If the head constraints and module identifiers of a rule can be omitted, the body constraints, BC, of that rule then constitute the background conditions. Section 4 illustrates the use of this inference mechanism with legal examples.

3.2.1 Case Representation

In this study, we regard a case as being a situation, that is, a simple set of anchored sentences. We give a sample case below, which is simplified from an actual legal precedent [42].

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 station. Tom was injured. Jim thought that Tom had died in the accident and left Tom on the street. Tom froze to death.

In $\mathcal{QUIXOTE}$ format, the aforementioned case contains objects, such as `mary`, `tom`, `jim`, `accident`, and `cold`, as well as several events, such as `abandon`, `find`, `make`, `injure`, `leave`, `death`, and `causes`. The relevance levels of these events are indicated through explicit attributes with ordering values.

```
&subsumption;;
l1 =< l2 =< l3;;

&rule;;
mary_case :: {mary, tom, jim, accident, cold,
   poor/[agent=mary, relevance=l1],
   abandon/[agent=mary, object=tom, relevance=l2],
   find/[agent=jim, object=tom/[state=crying], relevance=l1],
   accident/[agent=jim, relevance=l2],
   baby/[agent=tom, age=4months],
   injure/[agent=jim, object=tom, by=accident, relevance=l2],
   leave/[agent=jim, object=tom, relevance=l3],
   death/[agent=tom, cause=cold, relevance=l3]};;
```

In Mary's case, there were many interpretations on the responsibilities of the actions of Mary and Jim. For instance, a lawer reasoned that:

> "If Mary hadn't abandoned Tom, Tom wouldn't have died. In addition, the cause of Tom's death was not injury but freezing. Therefore there exists a causality between Tom's death and Mary's abandonment."

Another lawyer, however, argued differently:

> "There is a crime of Jim, for his abandonment of Tom. And in addition, Tom's death is indirectly caused by Jim's abandonment. Therefore, there exists a causality between Tom's death and Jim's abandonment."

These contradictory claims are documented, together with the final verdict as decided by the judge, as a judicial precedent. In the next subsection, we model these conflicting arguments with a set of *case rules*.

3.2.2 Case Rules

The deduction of legal arguments in $\mathcal{QUIXOTE}$ observes the following convention.

$$\overbrace{Head}^{\text{result}} \Leftarrow \overbrace{B_1, B_2, \cdots, B_n}^{\text{facts}} \,||\, \overbrace{Background\_conditions}^{\text{lawyer's interpretation}}.$$

Namely, B_i's in the above are the facts that were accepted by both the plaintiff and defendent beforehand, and the set of *Background Conditions* is the interpretation of causal relations between events, scopes of an agent's intention, and so on. For example, we can represent the following case-based rules in Mary's case (see Rule 2.2).

```
c >- cr1;;
cr1 :: responsible_for_injury/[agent=jim, to=tom]
        <=
       accident/[agent=jim],
       injury/[agent=tom]
       || { injury.cause=< accident};;
```

The first line, c >- cr1, claims that cr1 is an extended case of c, including the case rule. This rule claims: when there existed jim's accident and tom's injury as facts, and if the injury's cause was ascribed to the accident, jim is responsible to tom for the injury. Similarly, cr2 is another example of a case rule, again from Mary's case.

```
c >- cr2;;
cr2 :: responsible_for_protection_for_weak/[agent=jim, to=tom]
        <=
       accident/[agent=jim],
       baby/[agent=tom],
       injury/[agent=tom],
       leave/[agent=jim, object=tom]
       || { injury.cause=< accident};;
```

3.2.3 Induced Rules

The idea of an *induced rule* is to abstract some of ground terms in case rules, either by common sense knowledge or by legal theories. For example, if there are several similar accident cases, the attorneys may draw the following generalization, because the causality between the accident and the injury is agreed by both sides (refer to Rule 2.3):

```
ir1 :: responsible_for_injury/[agent=X, to=Y]
        <=
       accident/[agent=X],
       injury/[agent=Y] || { X =< person, Y =< person};;
```

In the above rule, restrictions on variables X and Y are given in the background conditions, such that they have to satisfy certain roles. The following ir2 is yet another, more abstract, form of ir1.

```
ir2 :: responsible/[agent=X, to=Y, for=Inj]
        <=
       Acc/[agent=X],
       Inj/[agent=Y, cause=Acc]
       || {Acc =< accident, Inj=<physical_damage,
       X =< person, Y =< person};;
```

In ir2, traffic accident and injury are abstracted to variable Acc and Inj, and are subsumed by their super concepts in the legal dictionary.

3.2.4 Legal Rules

Legal rules, or statutes, are formal sentences of codes. They are written in a form having free parameters. We illustrate this form using actual penal code (Japanese Penal Code, Article 199):

> "In case an intentional action of person A causes the death of person B and the action is not presumed to be legal, A is responsible for the crime of homicide."

The following is an example of representing the above in a *QUIXOTE* rule (see Rule 2.4):

```
lr1 :: responsible_for_homicide/[agent=A, to=B]
        <=
       Action/[agent=A],
       illegal/[act->Action],
       death/[agent=B, cause->Action]
       || {Action =< intend, A =< person, B =< person};;
```

In the description above, illegal/[agent=A, action -> Action] claims that the action Action done by A, such as self-defence, is not found to be legal. The statute for the legality of self-defence is described as follows (Japanese Penal Code, Article 38):

```
lr2 :: illegal/[act->Action]
        <=
       Action
       || {Action =< intend};;
```

The following statute (in Japan) states 'the responsibility for death by abandonment of the weak', that is:

> "If there is an intentional deed that corresponds to an abandonment of the weak, and the weak died because of the deed, the agent of the deed is responsible for the death."

```
lr3 :: responsible_for_death_by_abandonment_of_weak/[agent->W, to=Y]
        <=
       death/[agent=Y, cause=C],
       responsible_for_protection_for_weak/[agent=X, to=Y],
       weak/[agent=X],
       D/[agent=W, object=Y]
       || {D=<abandon, C=<D, D=<intend};;
```

Section 4.2 describes a legal reasoning example which involve `lr3` in its inference.

4 Query and Inference in $\mathcal{QUIXOTE}$

4.1 Constraints and Answer with Assumption

As mentioned in Section 3.2, $\mathcal{QUIXOTE}$ supports two kinds of constraints: *head constraints* and *body constraints*. During execution, the following transformation is performed first.

$$m_0 :: H \mid HC \Leftarrow m_1 : B_1 \mid C_1, \cdots, m_n : B_n \mid C_n \parallel Dot\_Cstr \cup Oterm\_Cstr;;$$
$$\Downarrow$$
$$m_0 :: H \mid \overbrace{HC \cup Oterm\_Cstr}^{head\ constraints} \Leftarrow m_1 : B_1, \ldots, m_n : B_n \parallel \overbrace{Dot\_Cstr \cup C_1 \cup \cdots \cup C_n}^{body\ constraints};;$$

That is,

- constraints (subsumption relation) on object terms ($Oterm\_Cstr$) are merged to the head constraint (HC), and are used as background conditions for the applicability of the rule,
- while constraints that includes a dotted term (Dot_Cstr) remain as the body constraint, and
- constraints on each object in the body (C_i) are merged into the body constraint.

To reply to a query, $\mathcal{QUIXOTE}$ often returns answer substitutions with a set of constraints among dotted terms called *assumptions*. An assumption is a set of unsatisfied constraints during derivation, such that they can be considered as being missing information. The control program or the user will then decide whether to fill in the missing information, or to invoke another query. Answering with assumptions is a kind of abduction reasoning and plays an important role in many AI applications, such as case-based reasoning systems and natural language processing. Except for constraints among dotted terms, $\mathcal{QUIXOTE}$ works like a conventional CLP language [24]. However, dotted term constraints in the body constraints work as assumptions if they are not satisfied in the head constraints. In this respect, $\mathcal{QUIXOTE}$ supports abductive queries to partial information databases, and such partiality differs from *incompleteness* in databases represented as null values or Skolem constants.

The formal derivation in $\mathcal{QUIXOTE}$ is explained as follows. Let G_m be a set of goals in the m-th stage of an execution, the next set of goals is derived from the rule $H|HC \Leftarrow B \parallel BC$:

$$G_{m+1} = (G_m - \{G\})\theta \cup B\theta.$$

where there is a most general unifier θ between H and G. Thus, the current goal $H\theta$ $(= G\theta)$ is removed from G_m, and new goals that are in the body part of the rule $B\theta$ are added. When $G_n = \phi$, execution ends. The conclusion is the set of resolved head constraints:

$$C_{m+1} = (C_m \cup HC)\theta$$

while a set of assumptions, or the remaining unsatisfied constraints, A_i, becomes:

$$A_{m+1} = (A_m \cup BC)\theta - C_{m+1}.$$

That is, A_i is the accumulation of body constraints $BC\theta$, some of them being removed from this accumulation when they are satisfied in HC $(= C_{m+1})$, and the final set of assumption, A_n, becomes the *abductive reason* for the conclusion.

As an example, consider the following piece of *QUIXOTE* code. It says that there is a crime and the judgement result is guilty if self-defence is illegal, but innocent if self-defence is legal.

```
case::crime;;
case::judgement[result=guilty] <= crime/[self_defence->illegal];;
case::judgement[result=innocent] <= crime/[self_defence->legal];;
```

The first clause tells us of the existence of an object, `crime`, but nothing about the properties of its `self_defence` attribute. The second clause means that if `crime` exists in the case and the `self_defence` property is subsumed by `illegal`, `judgement[result=guilty]` holds. When we initiate a query `?-case:judgement[result=Result]`, that is, the judgement result of the crime, *QUIXOTE* returns the following two independent answers.

```
Result=guilty   if   case:crime.self_defence=<illegal
Result=innocent   if   case:crime.self_defence=<legal
```

Each answer makes an assumption about the `self_defence` property of `crime` which comes from the body of the second or third clause. Neither constraint is satisfied by the head constraint, which is empty in this example, so they are accumulated as assumptions.

4.2 Inference of Legal Knowledge

We list a small case example (a traffic accident) and use it to show how the induced rule `ir1` is invoked.

```
n_case :: injure/[agent=tom];;
n_case :: traffic_accident/[agent=jim];;
```

```
&subsumption;;
traffic_accident =< accident;;
injure =< physical_damage;;
person >= {jim, tom};;

&submodule;;
ir2 -< n_case;;
```

Now, consider the following query to this program:

```
?- n_case : responsible/[agent=jim, to=X, for=Y].
```

This query may be read as "Is Jim responsible to someone, X, for something (represented variable Y)?" *QUIXOTE* returns the following answer.

```
** 1 answer exists **
** Answer 1 **
 IF n_case:injure.cause == traffic_accident THEN
    Y == injure
    X == tom
```

The answer says that if the cause of the injury is the traffic accident in this case, then Jim is responsible. Let us further consider a new case, where *QUIXOTE* invokes a sequence of case and legal rules to draw a conclusion, as shown in Figure 5. The new case, **hanako_case**, contains the following facts:

```
hanako_case ::{hanako, taro, jiro,
   death/[agent=taro, age=4months],
   baby/[agent=taro age=4months],
   injury/[agent=taro],
   abandon/[agent=hanako, object=taro],
   accident/[agent=jiro],
   leave/[agent=jiro, object=taro]};;
```

Using **Operation 3.2**, *QUIXOTE* would partially match **hanako_case** with **mary_case** (page 18) with the threshold relevance value, 12. That is, there is a rule substitution, θ, on cr2 (see Rule 2.5):

$\theta =$[hanako/mary, taro/tom, jiro/jim]

where 'x/y' stands for a substitution of y for x. The substituted rule, **cr2_s**, as generated, is represented as follows:

```
cr2_s :: responsible_for_protection_for_weak/[agent=jiro, to=taro]
          <=
         accident/[agent=jiro],
         baby/[agent=taro],
         injury/[agent=taro],
         leave/[agent=jiro, object=taro]
         || { injury.cause=< accident};;
```

The concept of *anchoring*, mentioned in Section 2.4, is realized in *QUIXOTE* by invoking either induced case rules or statutes within a case description. Let us suppose the following submodule relation:

```
&submodule;;
w >- hanako_case;;
w >- lr3;;
w >- cr2_s;;
```

with the following subsumption relations.

```
&subsumption;;
leave =< abandon;;
abandon =< intend;;
```

In addition, we need one more rule that is derived from common sense:

```
weak =< baby;;
```

With the query:

```
?-w:responsible_for_death_by_abandonment_of_weak/[agent=X, to=taro].
```

that means that "Is someone responsible for the death of Taro by abandoning the weak person?", *QUIXOTE* replies as follows.

```
** 2 answers exist **
** Answer 1 **
 IF w:injury.cause =< accident
    w:leave.agent >= responsible_for_death_by_abandonment_of_weak.agent
    w:death.cause =< leave
 THEN
    X =< jiro
** Answer 2 **
 IF w:injury.cause =< accident
```

```
    w:abandon.agent >= responsible_for_death_by_abandonment_of_weak.agent
    w:death.cause =< abandon
THEN
    X =< hanako
```

The first answer represents one interpretation of the causality in Hanako's case: if the cause of Taro's death is some event under Jiro's leaving Taro, then Jiro is responsible for the homicide. The second answer states yet another interpretation, that is, Hanako is responsible if Taro is killed by Hanako's intended abandonment. This rather confusing response arises from the fact that there were two deeds, `leave` and abandon, both of which can be regarded as being abandonment. That is, both belong to the same class in the legal dictionary. To verify this, one can further query the database with an additional constraint:

```
?-w:D || {D =< abandon}.
```

To which *QUIXOTE* replies:

```
?-w:D || {D=<abandon}.
** 2 answers exist **
** Answer 1 **
    D == leave
** Answer 2 **
    D == abandon
```

The chain of rules invoked in this example is shown in Fig. 6 (see also Figure 5).

Thus, in this section, we have shown that *QUIXOTE*:

- returns answers with assumptions when there are unsatisfied background conditions for applying legal and case rules,
- proposes all the alternative solutions to the query program for unsatisfied background conditions,
- accepts queries with additional information that has not yet been stored in its databases.

These features confirm the knowledge processing capability of *QUIXOTE* in supporting situated inference within an OODB framework, and to manage persistent data about case descriptions and legal rules.

5 Related Work

As with many system-level research projects, this reported work is strongly interdisciplinary. The technologies involved span many active research fields: deductive object-oriented databases, artificial

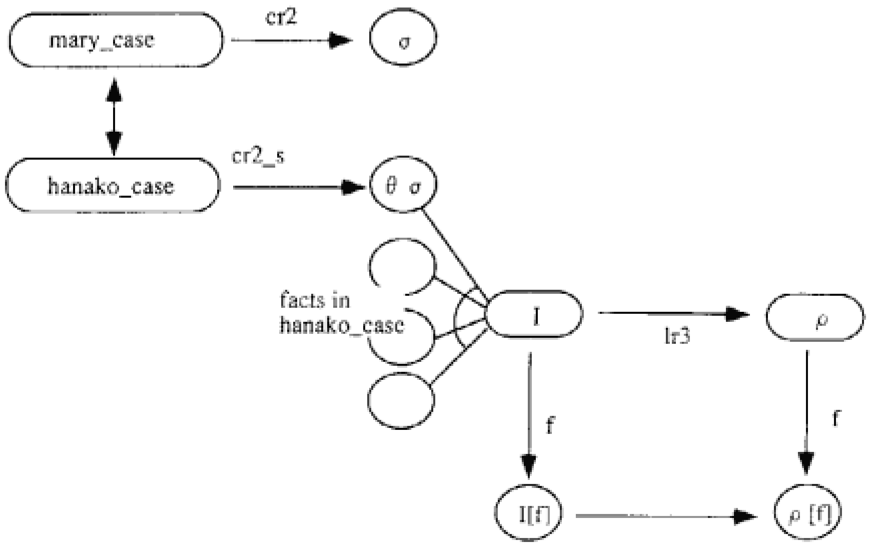

Figure 6: Situated inference graph in hanako_case.

intelligence and law, situation theory, and, to a lesser extent, constraint logic programming. The body of related work, in fact, covers such a wide area that this section mentions only the most closely related topics, and briefly at that. The references listed, however, would help the reader to probe further into individual areas of interests, or to further explore fruitful cross-fertilization of these fields.

- **Deductive Object-Oriented Databases**: To satisfy increasingly complex applications, we need to extend the expressive power and the processing capability of the current generation of database systems. The integration of deductive and object-oriented systems is recognized as a promising paradigm by the database community [14, 30, 10]. The pioneer work to capture complex data, such as complex objects and partial information, into logic programming has been done by COL [1] and $\mathcal{LDL}$ [41]. To cope with the problems, the early work introduces new constructors, such as tuples and sets, and related functions. Such extensions, however, fall short of providing many genuine deductive object-oriented features. $\mathcal{QUIXOTE}$ is one of the first implemented DOOD systems in which object-orientation concepts such as object identity, complex objects, type hierarchy, property inheritance, and the encapsulated method are merged seamlessly into a declarative logic programming with subsumption constraints. Due to space limitations, we are obliged to refer the reader to [62, 61] for a detailed discussion of the above. Currently, CHIMERA is another project persuing similar goal and approach [11].

  In line with the tradition of deductive database research [57, 36], the design of $\mathcal{QUIXOTE}$ places strong emphasis on theoretical foundations. (Object-oriented database research is generally empirical [29].) One key distinction of $\mathcal{QUIXOTE}$ from other work in this area is that its declarative semantics is based on ZFC$^-$/AFA, a hyperset theory proposed by Aczel [2, 60] which supports a nested structure and subsumption relations over complex objects, the latter being derived from the work of Barwise [9] and Mukai [38] on constraints over hypersets. The use of $\mathcal{QUIXOTE}$ to implement $\mathcal{SM}$ is largely motivated by the fact that its declarative semantics is closely related to the domain of natural language processing.

In addition, $\mathcal{QUIXOTE}$ uses a single language to achieve both query and programming. This makes the system inheritantly suitable for knowledge processing applications. At ICOT, we have been applying this database system to advanced applications in molecular biology, natural language processing, and legal reasoning [54, 55, 59].

- **AI and Law**: Broadly speaking, the field of AI and Law can be divided into two camps: experimental and formal. Researchers in the experimental camp develop computer programs with various blends of case-based and rule-based reasoning to address various legal problems while those in the formal camp are primarily concerned with theoretical or logical issues of legislation. Many AI-based programs have been implemented by the first camp. Notably, HYPO for trade secret law [4], part of a project on case-based reasoning conducted by Edwina Rissland and Kevin Ashley at the University of Massachusetts. CABARET [48], PROLEX [44], and GREBE [5] can apply either rule-based reasoning or case-based reasoning to top-level goals, rule antecedents, or matching of facts. Similarly, PROTOS [6] and CASEY [31] use general, rule-like knowledge to assess the similarity of cases.

  This work adopts the pragmatic approach of the experimental camp, that is, building knowledge programs that use both case-based and rule-based methods to reason about legal concepts. It differs, however, in two major aspects. First, existing programs have no database management capability, and, subsequently, each can experiment only with limited inference methods using a small sample of precedents. This work proposes to remedy such limitations by integrating an advanced database system into the overall knowledge system architecture. Secondly, existing programs often lack a rigorous formulation for representing and analyzing legal knowledge, such as the interpretation of open-textured concepts and legal inference under different situations. In contrast, the design of our legal knowledge bases is firmly based on the $\mathcal{SM}$ model, which has precise definitions of legal concepts and rules, and their associated operations. In this respect, we are closer to the formal camp.

  A representative group of the formal camp is the legal reasoning group at Imperial College, London [34, 52], who advocate the logical programming approach to study the language of legislation, such as the British Nationality Act. Since then, many legal reasoning researchers have expanded this approach with many types of exotic logics covering the temporal, deontic, and abduction aspects of law [26, 22, 45]. Few, however, have carried the formulation into practical implementation. In contrast, $\mathcal{SM}$ takes the formal approach of situation theory, and this formulation is used to design subsequent legal knowledge bases. In Section 2, we showed that situation semantics is a more natural way to interpret open-texture concepts and to represent legal inference. The two approaches, however, are not orthogonal. By treating a situation as a set of infons, we can incorporate many established techniques of classical logic into $\mathcal{SM}$ [57].

- **Situation Theory**: Situation theory is a recent extension of classical Tarskian logic [35], especially for the study of the philosophy of language and computational linguistics. This field, however, is still evolving, and there are many alternatives in interpreting its basic concepts, such as situations and support relation. To avoid ambiguity, we must distinguish abstract situations for formulation and computation from real situations that were structured parts of the real world [15]. The mapping from situations to infons was first proposed by Fernando [20] as $s \models_\Delta \sigma$ iff $\sigma \in \Delta s$, where Δ is a *support-map* function, from a situation (a structured part of world) to a set of infons. Based on the properties of the legal cases, in $\mathcal{SM}$, we also defined situations as sets of infons, similar to that of Devlin's, and adopted the latter's definition for types, parameters, and anchoring [15, 16].

  Many features of situated inference were discussed by Barwise and Devlin [9, 16]. Of these, two key features are incorporated in our situated inference framework. One is the feature of *involves* relation ('$\Rightarrow$') [7], which we used to denote legal deduction: $s \models \sigma$ and $\sigma \Rightarrow \tau$, then $s \models \tau$. The other is the use of constraints as background conditions: $H \Leftarrow B \parallel C \quad \text{iff} \quad H \cap C \Leftarrow B$.

  Kowaski and Sergot proposed extending the logic programming approach for situation or event calculus [33, 32]. The basic idea is to use a metapredicate *demo(T,P)* which holds when a sentence named P can be proved (or demonstrated) from the theory about a situation or an event named T. This approach, however, requires the representation of two levels of knowledge, object theory and metalevel theory, in computer programs. Further, it does not intend to support many of the key features of situation theory, and involves additional programming efforts, and thus computational overhead, to do so. Though not a database language, PROSIT, proposed by Nakashima [39, 40], is one of the first programming languages for situated inference. In PROSIT, every rule is asserted in a hierarchical situation, in which rules can be inherited. $\mathcal{QUIXOTE}$ offers the same modeling ability, and, further, it extends the simple infons in PROSIT to complex objects. The concept of complex object of $\mathcal{QUIXOTE}$ inherits PST (Partially Specified Term) of CIL [37], which realizes the partiality of information of situation theory. Although any infon is considered as being partially specified information in CIL, for legal reasoning applications, we adopt Devlin's R^n definition [15]. That is, every relation R can define its maximal set of n parameters, and an infon specifies complete information.

To date, situation theory has barely been used in actual applications. Rounds' application to relational databases [49] seems to be the only example in the database area. That work, however, showed only a theoretical aspect and is not an actual application. This differs from our pragmatic goal of building large, advanced legal reasoning systems using situation theory and deductive object-oriented databases.

- **Constraint Logic Programming:** Constraint solving is a paradigm that searches for a solution to satisfy the global consistency of an entire program, from a combination of locally defined constraints. Constraint logic programming (CLP) language is an extension of logic programming language with this constraint-solving paradigm. $\mathcal{QUIXOTE}$ is one such language, together with Prolog-III [13], CLP($\mathcal{R}$) [23] that is an instance of CLP(X) [24, 25], CAL, [3], and cu-Prolog, the predecessor to $\mathcal{QUIXOTE}$ [56]. All of these deal with equalities or inequalities between numbers, lists, and other domains as constraints. The distinctive feature of $\mathcal{QUIXOTE}$ is that it has only one domain, that is, objects, and the constraints are the subsumption relations between objects.

  The constraints themselves do not specify the order of execution. Thus, the delayed evaluation, a control mechanism to suspend a part of program until all variables of that part are bound, becomes the most important problem in computation. The current CLP topic of great interest is the execution of control concurrently. $\mathcal{QUIXOTE}$ does not deal with concurrent constraints directly. It is, however, implemented in the concurrent logic programming language, KL1 [12], such that constraints could be executed concurrently at the operating system level.

6 Conclusions

This work is primarily theoretical but has a concrete application in an interdisciplinary field, AI and Law. To the best of our knowledge, this is the first reported work that brings together two previously unrelated fields, namely, deductive object-oriented databases and situation theory, to design knowledge systems for solving complex problems and for modeling human intellectual behavior. It is also the first attempt to enhance the reasoning capability and application scale of the current generation of legal reasoning systems with an advanced database tool.

In this paper, we have outlined the motivation behind this study, presented the basic features of a formal model for legal reasoning, and a deductive object-oriented database for implementing this model. The foundation of this model, $\mathcal{SM}$, is based on the theory of situations and clearly defines the notions of open-texture concepts and situated inference in the legal domain. The purpose of this model is to study the fundamental issues of AI and Law at the abstraction level, to help design better and more robust legal reasoning systems. In Section 2, we introduced the key features but leave a more detailed description in a future paper. In Sections 3 and 4, we described how $\mathcal{QUIXOTE}$, a deductive object-oriented database system, is used to implement $\mathcal{SM}$ for our legal reasoning applications. In addition, we have illustrated the features of $\mathcal{QUIXOTE}$ with implemented legal examples.

$\mathcal{QUIXOTE}$ provides a single language for both query and programming purposes, and it exhibits the inference features of deduction, object-oriented, and constraint logic programming. Most legal reasoning systems are small programs that lack the database management capability to access and store large volumes of data, presenting a stumbling block to the growth of this knowledge-intensive field. The DOOD approach is proposed here to satisfy such needs. In addition, research into legal reasoning systems is closely related to a broader and more complex field, natural language processing (NLP). The ability of DOOD systems, such as *QUIXOTE*, to model abstract concepts of situation theory in a database environment may pave the way for the NLP community to tackle concrete, demanding problems, such as building a comprehensive dictionary database of general linguistic concepts.

An earlier version of the KL1-based control program is to draw all alternative conclusions about a legal goal, in parallel, from cases and statutes [42]. Currently, the legal reasoning group at ICOT is enhancing the control program with the capability to engage in legal debate, based on queries made to the underlying legal knowledge bases. The computational model of this program is described in [43, 58]. In short, the argumentation process of this model can be likened to a two-agent game. One agent puts forward an argument. The other agent recognizes the situation, generates candidates to refute the claim, and selects the best one for the next move. The game ends when any one agent can no longer make a move, that is, all possible alternatives have been exhausted. During the debate, both agents would query the underlying set of legal knowledge bases extensively to obtain sufficient information for their next moves. On the other hand, the current *QUIXOTE* system is implemented using the parallel logic programming language, KL1, on the parallel inference machine, PIM. Both were developed as part of the FGCS by ICOT. The post-FGCS project is working on porting *QUIXOTE* from the current platform to the more generally accessible Unix and C environments.

Acknowledgements

This work is a part of the Fifth Generation Computer Systems project of Japan and its follow-on project. The authors would like to thank the members of the knowledge base management systems group and the legal reasoning group at ICOT, especially, K. Nitta, K. Yokota, H. Yasukawa, and H. Tsuda. Further, we would also like to acknowledge the leadership of K. Fuchi and S. Uchida.

Affliation of Author

Stephen T. C. Wong, ICOT, 21 F., Mita-Kokusai Building, 4-28, Mita 1, Minato-Ku, Tokyo, 108, Japan

Satoshi Tojo, Mitsubishi Research Institute, 8-1 Shimomeguro 1, Meguro-ku, Tokyo 153, Japan.

References

[1] S. Abiteboul, S. Grumbach, "COL: A logic-based language for complex objects," *Proc. Int. Conf. on Extending Database Technology*, in Lecture Notes of Computer Science 330, Springer, 1988.

[2] P. Aczel, "Non-well founded set theory," CSLI Lecture Notes, No. 14, 1988.

[3] A. Aiba, K. Sakai, Y. Sato, D.J. Hawley, and R. Hasegawa, "Constraint logic programming language CAL," *Proc. of the International Conference on Fifth Generation Computer Systems*, Tokyo, 1988.

[4] K. Ashley, *Modeling legal argument*, MIT Press, Cambridge, MA, 1990.

[5] L.K. Branting, "Building explanations from rules and structured cases," *International Journal of Man-Machine Studies*, Vol. 34, No. 6, June 1991. pp. 797-838.

[6] E.R. Bareiss, *Protos: A unified approach to concept representation, classification, and learning*, PhD thesis, The University of Texas at Austin, 1988.

[7] J. Barwise, J. Perry, *Situations and attitudes*, Cambridge, MA, MIT Press, 1983.

[8] J. Barwise, *The situation in logic*, CSLI Lecture Notes 17, Stanford, CA, 1988.

[9] J. Barwise, "AFA and the unification of information," in *The situation in logic*, CSLI, Lecture Notes 17, 1988.

[10] S. Ceri, G. Gottlob, L. Tanca, *Logic programming and databases*, Springer-Verlag, Berlin, 1990.

[11] S. Ceri, R. Manthey, "Overview of CHIMERA – The conceptual interface of IDEA," ICOT Lecture, Jan. 12, 1993.

[12] T. Chikayama, "Operating system PIMOS and kernel language KL1," *Proc. Int. Conf. of Fifth Generation Computer Systems*, ICOT, Tokyo, June, 1992, pp. 73-88.

[13] A. Colmerauer, "Opening the Prolog III Universe: A new generation of Prolog promises some powerful capabilities," *BYTE*, August, 1987, pp. 177-182.

[14] C. Delobel, M. Kifer, and Y. Masunaga (eds.) *Deductive and object-oriented databases: Prof. 2nd Int. Conf. on Deductive and Object-Oriented Databases (DOOD91)*, LNCS 566, Springer, 1991.

[15] K. Devlin, "Infons and Types in an Information-Based Logic," in R. Cooper and K. Mukai and J. Perry (eds.), *Situation Theory and its Applications, Vol. 1*, pp. 79-95, 1990.

[16] K. Devlin, *Logic and information I*, Cambridge University Press, 1991.

[17] M. Dincbas, P. Van Hentenryck, H. Simonis, A. Aggoun, T. Graf, and F. Berthier, "The Constraint Logic Programming Language CHIP," *Proc. of the International Conference on Fifth Generation Computer Systems*, Tokyo, 1988.

[18] D. Dowty, R. Wall, and S. Peters, *Introduction to Montague Semantics*, D. Reidel, 1981.

[19] G. Fauconnier, *Espas Mentaux*, Editions de Minuit, 1984.

[20] T. Fernando, "Infons and types in an information-based logic," in R. Cooper, K. Mukai, and J. Perry, *Situation Theory and its Applications, Vol. 1*, 1990, pp. 97-116.

[21] A.v.d.L. Gardner, *An artificial intelligence approach to legal reasoning*, MIT Press, Cambridge, Massachusetts, 1987.

[22] T. Gordon, "An abductive theory of legal issues," *International Journal of Man-Machine Studies*, Vol. 35, No. 1, 1991, pp. 95-118.

[23] J. Jaffer, J-L. Lassez, *Technical Report: constraint logic programming*, IBM Watson Research Center, 1986.

[24] J. Jaffer, J-L. Lassez, "Constraint logic programming," *Proc. 4th IEEE Symposium on Logic Programming*, 1987.

[25] J. Jaffer, J-L. Lassez, "Methodology and implementation of a CLP system," *Proc. Fourth International Conference on Logic Programming*, 1987, pp. 858-876.

[26] A. Jones, M. Sergot, "Deontic logic in the representation of law: towards a methodology," *AI and Law*, Vol. 1, No. 1, 1992, pp. 45-64.

[27] H. Kamp, " A theory of truth and semantic representation," in J. Groenendijk, T. Jansson, and M. Stockhof, (ed.), *Methods in the Study of Language Representation*. Math Carter, Amsterdam, 1981.

[28] M. Kifer, G. Lausen, "F-logic – a higher order language for reasoning about objects, inheritance, and schema," *Proc. ACM SIGMOD Int. Conf. on Management of Data*, Portland, June, 1989, pp. 134-146.

[29] W. Kim, "Object-oriented databases: definition and research directions," *IEEE Trans. on Knowledge and Data Engineering*, Vol. 2, No. 3, Sept., 1990, pp. 327-341.

[30] W. Kim, J.M. Nicolas, and S. Nishio (eds.) *Deductive and object-oriented databases: Proc. the First International Conference on Deductive and Object-Oriented Databases (DOOD89)*, North-Holland, 1990.

[31] P. Koton, *Using experience in learning and problem solving,* PhD thesis, Department of Electrical Engineering and Computer Science, MIT, 1988.

[32] R. Kowaski, "Problems and promises of computational logic," Technical report, Imperial College, Department of Computing, London, September, 1990.

[33] R. Kowaski, M. Sergot, "A logic-based calculus of events," *New Generation Computing,* Vol. 4, No. 1, pp. 67-95.

[34] R. Kowaski, M. Sergot, "The use of logical models in legal problem solving," in A. Narayan and M. Bennun (eds.) *Law, Computer, and Artificial Intelligence*, Ablex, 1990.

[35] E. Mendelson, *Introduction to mathematical logic*, 3rd ed., Wadsworth & Brooks/Cole, 1987.

[36] J. Minker (ed.) *Foundations of deductive databases and logic programming,* Morgan Kaufmann, Los Altos, CA, 1988.

[37] K. Mukai, H. Yasuakwa, "Complex indeterminates in prolog and its application to discourse models," *New Generation Computing,* Vol.3, No.4, 1985, pp. 441-466.

[38] K. Mukai, "CLP(AFA): Coinductive semantics of horn clauses with compact constraint," *Proc. Second Conf. on Situation Theory and Its Applications*, Kinloch Rannoch, Scotland, Sept. 1990.

[39] H. Nakashima, H. Suzuki, P. Halvorsen, and S. Peters, "Towards a computational interpretation of situation theory," *Proc. of International Conference on Fifth Generation Computer Systems '88,* 1988, pp. 489-498.

[40] H. Nakashima, S. Peters, and H. Schüzte, "Communication and inference through situations," *Proc. 12th International Joint Conference on AI*, 1991.

[41] S. Naqvi, S. Tsur, *A logical language for data and knowledge,* Computer Science Press, MD, 1989.

[42] K. Nitta, Y. Ohtake, S. Maeda, M. Ono, H. Ohsaki, and K. Sakane, "HELIC-II: A legal reasoning system on the parallel inference machine," *Proc. Int. Conf. of Fifth Generation Computer Systems*, ICOT, Tokyo, June, 1992, pp. 1115-1124.

[43] K. Nitta, S. Wong, Y. Ohtake, "A computational model for trial reasoning," *Proc. 4th Int. Conf. on AI and Law*, Amsterdam, June, 1993.

[44] R.F. Walker, A. Oskamp, J. Scjroclx. G. Van Opdorp, and P. Van Den Berg, "Prolexs, creating law and order in a heterogeneous domain," *International Journal of Man-Machine Studies*, Vol. 35, No. 1, 1991, pp. 35-68.

[45] D. Poulin, E. Mackaay, P. Bratley, J. Fremont, "Time server - a legal time specialist," in A. Martino (ed.) *Expert Systems in Law*, 1992, pp. 295-312.

[46] E.L. Rissland, (ed.), Special issue: AI and Legal Reasoning, Part 1, *International Journal of Man-Machine Studies*, Vol. 34, No. 6, June 1991.

[47] E.L. Rissland, (ed.), Special issue: AI and Legal Reasoning, Part 2, *International Journal of Man-Machine Studies*, Vol. 35, No. 1, July 1991.

[48] E.L. Rissland, D.B. Skalak, "CABARET: statutory interpretation in a hybrid architecture," *International Journal of Man-Machine Studies*, Vol. 34, 1991, pp. 839-887.

[49] B. Rounds, "Situation-theoretic aspects of databases," in J. Barwise, J. M. Gawron, G. Plotkin, and S. Tutiya (eds.), *Situation Theory and its Applications, Vol. 2*, 1991, pp. 229-255.

[50] M. Sergot, "The representation of law in computer programs: a survey and comparison," in T.J.M. Bench-Capon (ed.), *Knowledge Based Systems and Legal Applications*, Academic Press, 1991, pp. 3-68.

[51] M. Sergot, F. Sadri, B. Kowalski, F. Kriwaczek, P. Hammond, and H. Cory, "The British Nationality Act as a Logic Program," *Communication of the ACM*, Vo. 29. No. 3, 1986, pp. 370-386.

[52] K. Taki, "Parallel inference machine PIM," *Proc. International Conference on Fifth Generation Computer Systems*, ICOT, Tokyo, June, 1992, pp. 50-72.

[53] H. Tanaka, "Integrated system for protein information processing," *Proc. Int. Conf. on Fifth Generation Computer Systems*, ICOT, Tokyo, June, 1992, pp. 321-329.

[54] S. Tojo, H. Yasukawa, "Situated inference of temporal information," *Proc. Int. Conf. on Fifth Generation Computer Systems*, ICOT, Tokyo, June, 1992, pp. 395-404.

[55] H. Tuda, "cu-Prolog for Constraint-Based Grammar," *Proceedings of International Conf. Fifth Generation Computer Systems*, Tokyo, June, 1992, pp. "347–356".

[56] J.D. Ullman, "Implementation of logical query languages for databases," *ACM Trans. on Database Systems*, Vol. 3, No. 3, 1985.

[57] S. Wong, "A situation-theoretic model for trial reasoning," *Proc. of the 6th Int. Symp. on Legal Knowledge and Legal Reasoning Systems*, Tokyo, Oct., 1992, pp. 32-54.

[58] N. Yamamoto, "TRIAL: a legal reasoning system," *Proc. of Joint French-Japanese Workshop on Logic Programming*, Renne, France, July, 1991.

[59] H. Yasukawa, K. Yokota, "Labeled graphs as semantics of objects," *Proc. SIGDBS and SIGAI of Information Processing Society of Japan*, Oct., 1990.

[60] K. Yokota, H. Yasukawa, "Towards an integrated knowledge base management system," *Proc. Int. Conf. on Fifth Generation Computer Systems*, ICOT, Tokyo, June 1-5, 1992, pp. 89-112.

[61] K. Yokota, H. Tsuda, Y. Morita, S. Tojo, H. Yasukawa, "Specific features of a deductive object-oriented database language $\mathcal{QUIXOTE}$," *Proc. Workshop on Combining Declarative and Object-Oriented Databases*, ACM SIGMOD, Washington, D.C., May 29, 1993.